Abstract

We present the SIXTE software package, a generic, mission-independent Monte Carlo simulation toolkit for X-ray astronomical instrumentation. The targets to be observed are stored in so-called SIMPUT files implementing a comprehensive format to describe simple as well as sophisticated source models.
Based on such a source definition, a sample of photons is produced as input for an instrument simulator. The software toolkit contains modules for imaging X-ray telescopes, collimators, and different detector types in order to assemble an instrument model. The flexibility achieved by this approach makes it a powerful tool for the development and exploration of new instruments. The implementation of specific effects enables the analysis of characteristic features such as pile-up and their impact on observations.

Change Record

Issue	*Date*	*Description of Change*	*Affected Pages*
1	2016-04-17	pre-Release	All
2	2016-04-26	refined the tutorials	21–66
3	2017-07-24	updated erosita tutorial	62–66
4	2017-10-10	updated background and tutorial	16–end
5	2018-01-29	added tessim tutorial	62
6	2018-01-29	revised tutorials, added appendix	34–71
7	2018-12-22	added time variability tutorials	45–49
8	2019-01-04	added description of PHA2PI correction	33-35,78,79
9	2020-11-25	updated WFI instrument and tutorial	17-19,44-47,56-66
10	2021-02-01	distinction between calorimeter simulators	31
11	2021-11-17	updated WFI and X-IFU instrument files	18,45–48,57,62–67,74–79
		updated background description	15,16
12	2022-03-25	Smaller fixes for clarity	52,55,60,62,65–67,71,72,75
13	2024-10-21	Updates for SIXTE version 3	All

Distribution List

Organization	*Name*	*Organization*	*Name*	*Organization*	*Name*

Approvals

Function	*Name*	*Date*	*Signature*
Author	T. Dauser		N/A
Author	J. Wilms		N/A
Author	P. Peille		N/A
Author	O. König		N/A
Author	C. Kirsch		N/A
Author	M. Lorenz		N/A
Author	L. Michalski		N/A

Contents

List of Figures

List of TBD Issues

List of TBC Issues

Documentation

Reference Documents

RD1	*Athena*+ Response Files (Brand, T.)	ECAP-ATHENA+-201402102

1 Introduction to the SIXTE Simulator

Computer simulations play an important role in the development of future X-ray missions as well as in the scientific analysis of observations with existing facilities. On the one hand they provide the possibility to investigate the interdependence of various instrumental effects and on the other hand they can produce test data for different purposes as a substitute for real data.
A large number of existing simulation tools address the particular needs of individual instruments. Examples are `MARX` (Wise et al., 1997) for *Chandra* (Weisskopf et al., 2002), `SciSim` (Gabriel et al., 2005) for X-ray Multi-Mirror Mission Newton (*XMM-Newton*, Jansen et al., 2001), or `NuSim` (Madsen et al., 2011; Zoglauer et al., 2011) for the Nuclear Spectroscopic Telescope ARray (*NuSTAR*, Harrison et al., 2010). Traditionally, however, these software packages are restricted to the particular mission for which they were developed. The only exception is `simx`[1], developed by R.K. Smith (CfA), which enables the fast simulation of observations for several X-ray telescopes. The latter software package, however, does not allow the detailed investigation of timing-related features and includes only a limited range of detector effects. While very useful to get a fast overview of the capabilities of an instrument, `simx` by design is not well suited for more detailed simulation studies of an instrument's scientific performance.
In this manual we describe the generic Monte Carlo code SImulation of X-ray TElescopes (SIXTE, Dauser et al., 2019). In addition the SIMPUT format and its functionalities are described, which are used to specify the SIMulation inPUT. By providing a sophisticated detector model combined with an in-depth description of the source physics, the SIXTE framework can be used both to investigate specific features of various X-ray instruments – including simulations in preparation of observations – and to generate test data for the development of new scientific facilities. These tasks are performed with small computational effort but sufficient physical accuracy. Current applications are extended Röntgen Survey with an Imaging Telescope Array (eROSITA) (Predehl et al., 2006, 2007, 2010a,b, 2011; Predehl, 2012) aboard Spectrum-Roentgen-Gamma (*SRG*) (Pavlinsky et al., 2006, 2008, 2009), the Large Area Detector (LAD) (Zane et al., 2012) and the Wide Field Monitor (WFM, Brandt et al., 2012) aboard *LOFT* (Feroci et al., 2010, 2012; Schmid et al., 2012), the HXI of the MIRAX experiment (Braga & Mejía, 2006; Grindlay et al., 2011) for the Lattes satellite, and the Wide Field Imager (WFI) (Rau et al., 2013) and the X-IFU (Barret et al., 2013) aboard Advanced Telescope for High ENergy Astrophysics (*Athena*) (Nandra et al., 2013). SIXTE was also used (Schmid et al., 2010, 2011) for the assessment studies of the International X-ray Observatory (IXO, Bookbinder, 2010; Bookbinder et al., 2010; Barcons et al., 2011a) and *Athena* (Barcons et al., 2011b).
The manual presented below relies heavily on Dauser et al. (2019) from which it takes parts verbatim.

1.1 General Idea

The complex idea of a general observation simulation is broken down into three major tasks (Fig. 1): At first, from a flexible source description in SIMPUT a number of photons are generated. In the second step, these photons are propagated through the representation of the optics, resulting in a list of impact times, positions and energies. In the third step, the impacts cause a signal in the virtual model of the detector, which simulates complex read-out behaviour and outputs an event list. These steps are covered in more detail in the following sections.
SIXTE enables Monte Carlo simulations based on individual photons, which are produced for a selected source type and processed through a modular instrument model. The instrument model can be assembled flexibly from multiple components and is therefore applicable to various different X-ray and gamma-ray instruments. Its parameters are not defined in the source code but in a particular eXtensible Markup Language (XML) file such that in many cases the configuration can be adapted without modifying the program.

[1] `http://hea-www.harvard.edu/simx/`

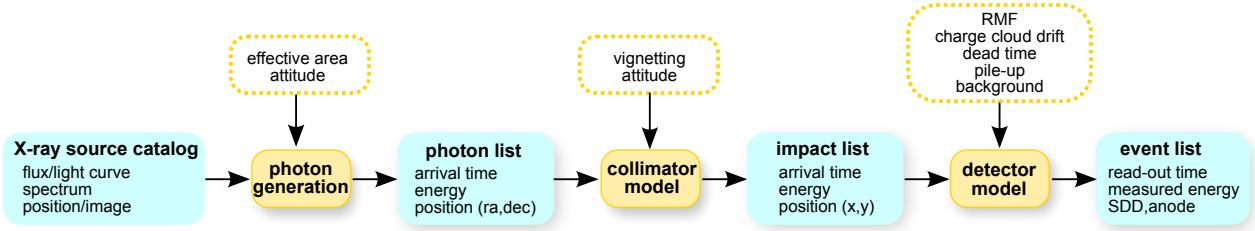

Figure 1: This flow chart illustrates the three major parts of the SIXTE simulation software (yellow boxes) needed to create an event list based on a X-ray source catalog. The dashed boxes above each simulation step specify the input data needed by these routines.

In general, the SIXTE software is designed such that it can simulate any photon based detector, as long as a description of its response is available.
The software package enables physically accurate simulations with a manageable computational effort. Where it is appropriate, complex physical processes are implemented as simplified models based on the respective calibration files instead of using computationally challenging ray-tracing algorithms. For instance, the imaging process in a Wolter telescope is performed via a randomization process using the PSF(Point Spread Function) of the telescope (although, if needed, a ray-tracer could easily be included in the package). The energy resolution of a particular detector is modeled by its Redistribution Matrix File (RMF, George et al., 1998, 2007). However, the selected implementation also allows an analysis of very specific effects such as pile-up, which require a detailed description of the detector operation. Comparisons with measured data show that the simulation produces realistic results (Martin, 2009; Schmid, 2012). A detailed description of SIXTE is given by Schmid (2012).

1.2 The Software

The functionality of the simulation code is contained in the two separate software packages SIMPUT and SIXTE. Their source code is maintained with the version control system `git`[2]. All software is freely available and licensed under the GNU General Public License[3].
The SIMPUT package [4] comprises the SIMPUT library with the basic routines to access SIMPUT files and to produce photons for the specified targets, as well as a set of tools to handle the generation and management of SIMPUT files. As shown in A, these tools allow, e.g., the easy inclusion of an `Xspec`[5] (Arnaud, 1996) spectral model in a SIMPUT file. The library includes all relevant routines to interface to the SIMPUT format and can be used, e.g., in the development of more specialized X-ray instrument simulations.
The SIXTE package[6,7] uses the SIMPUT library for the photon generation process, but also contains its own library, which implements the setup and handling of instrument models. The package provides the actual simulation tools for the generic detector model and for multiple specific instrument models. An example of the usage of these tools is shown in B.
The majority of the code is implemented in `C` or `C++` and can be compiled and installed using `cmake` [8]. Data are stored in Flexible Image Transport System (FITS) files using the `cfitsio` library (Pence, 1999). In particular

[2] `http://git-scm.com/`
[3] `http://www.gnu.org/licenses/gpl-3.0`
[4] `http://www.sternwarte.uni-erlangen.de/git.public/simput.git/`
[5] `http://heasarc.nasa.gov/xanadu/xspec/`
[6] `http://www.sternwarte.uni-erlangen.de/research/sixte/`
[7] `http://www.sternwarte.uni-erlangen.de/git.public/sixt/`
[8] https://cmake.org/

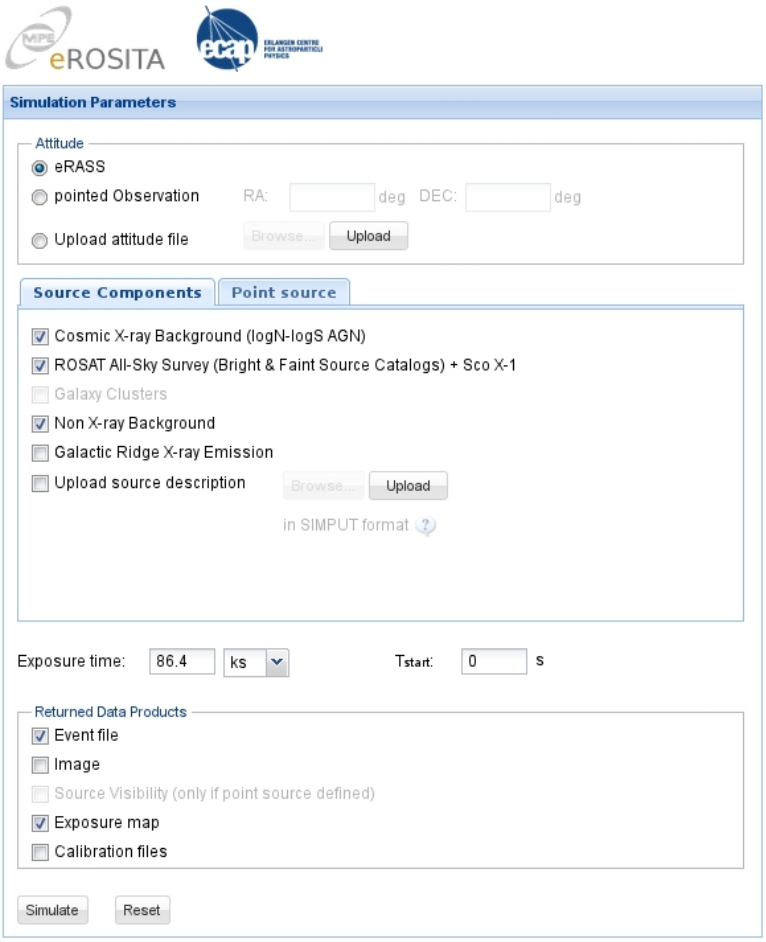

Figure 2: Web interface for simulating observations with eROSITA accessible at `http://cetus.sternwarte.uni-erlangen.de/~erosim/`. The user can set up the observation based on a set of parameters and select the observed targets or upload an own SIMPUT file. After the simulation is run on the server at the ECAP, the user can download the selected data products such as event files or an image.

the instrument calibration files follow the standards of the High Energy Astrophysics Science Archive Research Center (HEASARC)[9]. The interface of the implemented tools is designed similarly to the `ftools`[10] using the Parameter Interface Library (PIL, Borkowski et al., 2002) or All Purpose Parameter Environment (APE) to read program parameters and other functionality of the High Energy Astronomy software (HEASOFT) package. Physical units in the FITS files are specified according to George & Angelini (1995).

SIXTE routines can also be used to power the backend of a web interface. This option allows remote as well as easy access to simulated observations of popular targets with a predefined instrument configuration. We provide a number of web interfaces for different X-ray missions on a server at the Erlangen Centre for Astroparticle Physics (ECAP). As an example, in Fig. 2 a screenshot of the web interface for eROSITA is displayed.

The necessary memory and CPU power to run SIXTE strongly depends on the simulated instrument, the complexity of the source field, and the duration of the observation. Generally, simple simulations of observations with a few hours length do not require more than 1 GByte of memory and run within a few minutes on a normal desktop computer. Simulations can be easily run in parallel, by splitting the observation in multiple parts and simulating those independently.

[9] `http://heasarc.gsfc.nasa.gov/docs/heasarc/caldb/caldb_doc.html`
[10] `http://heasarc.nasa.gov/docs/software/ftools/`

2 Source Definition (SIMPUT)

In order to enable realistic simulations of astrophysical observations, which are needed to prove that a given instrument can fulfill a mission's scientific requirements, suitable specifications of the observable sources are essential. For this purpose we have developed the SIMPUT file format (Schmid et al., 2013)[11]. It is based on the Flexible Image Transport System (FITS, Wells et al., 1981; Ponz et al., 1994; Hanisch et al., 2001; Pence et al., 2010)[12] and makes use of the specification of data in different Header and Data Units (HDUs).
The main advantage of SIMPUT is an instrument-independent definition of sources such that the same underlying data can be used for simulations of different instruments. Depending on the knowledge or assumptions about a particular source, either basic or sophisticated models can be constructed with phenomena ranging from simple energy spectra to spatial variations of the observed spectra. Another key element is the possibility to specify time variability.
The use of the SIMPUT format is not restricted to SIXTE, but it is designed for general application. For instance, the simulation software `simx` implements it as an alternative input format. In the following we give a brief summary of the basic characteristics of the format. The full formal description of the format is given in the relevant standard document (Schmid et al., 2013).

2.1 Source catalog

The core of a SIMPUT file is a source catalog containing one or multiple X-ray sources. It consists of a table with entries for each object defining its basic properties such as its position and the observed flux in a particular energy band. In addition, references to other HDUs are given, which contain more detailed information about the energy spectrum, time variability, and spatial extent of the source. While the reference to a spectrum is obligatory for each source, the specification of time variability and spatial extent are optional. The use of references to spectra and other source characteristics allows their reuse in large source catalogs, as is needed, e.g., when simulating deep X-ray fields or a sky survey. A schematic layout of a sample catalog is illustrated in Fig. 3.

2.2 Energy Spectrum

The most common way to define a spectrum in SIMPUT is the specification of the photon flux density $P(E)$ distribution in the units photons $\mathrm{s}^{-1}\,\mathrm{cm}^{-2}\,\mathrm{keV}^{-1}$. Alternatively the spectral shape can be modeled by a sample of photons stored in an HDU table, containing simply the photon energy and arrival time. The latter approach is convenient if the source model is based on input, e.g., from a hydrodynamical simulation, which often provides a list of emitted photons.
An individual spectral model can be assigned to a variety of sources with different brightness but the same spectral shape. Therefore only the shape but not the normalization of a spectral definition is taken into account and the actual flux of each individual source is determined by the reference flux assigned to this source in the catalog HDU.

2.3 Time Variability

Time variability can be defined by a light curve, a Power Spectral Density (PSD), or a photon list with time information assigned to each photon. Light curves are either continuous or periodic and describe the variation of the source brightness. They are quite suitable to model, e.g., phenomena with a typical time evolution such as type-I X-ray bursts or periodic signals from a pulsar. PSDs are convenient to describe random variations such as

[11] SIMPUT version 1.1.0 discussed in the following. A detailed descripton is available at `http://hea-www.harvard.edu/HEASARC/formats/simput-1.1.0.pdf`

[12] `http://fits.gsfc.nasa.gov/`

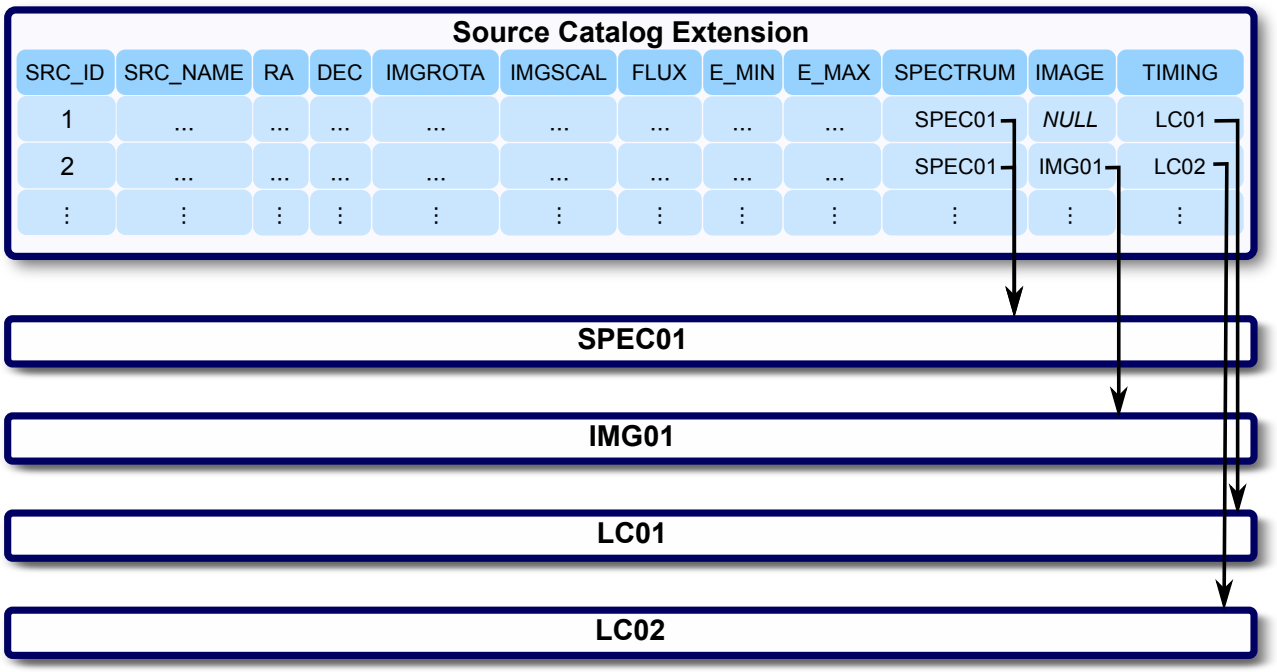

Figure 3: Schematic layout of a SIMPUT file. The source catalog contains the main characteristics of one or multiple sources, such as their positions and observed fluxes. Additional data describing energy spectra, spatial extent and time-variability, are stored in separate HDUs and linked in the source catalog.

noise. In order to obtain a proper time distribution of simulated photons, they can be internally converted to light curves by the simulation software using, e.g., the algorithm introduced by Timmer & König (1995). Photon lists are often convenient to use if input is available from a hydrodynamical simulation of the X-ray source.

2.4 Spatial Extent

In contrast to point-like sources, which are only specified by their position as stated in the source catalog, extended sources can be defined with an additional FITS image representing the spatial distribution of the observed flux or a photon list including spatial information. The same image can be used for multiple sources, scaled and rotated differently by the respective parameters in the SIMPUT source list (see SIMPUT documentation for more details). Both kinds of data can be obtained, e.g., from an observation or a simulation of the source.

3 Photon Generation

Based on the definition in the SIMPUT file, a sample of photons is produced with the following three basic characteristics required for the subsequent instrument simulation: the time, energy, and direction of origin of each photon.

3.1 Source Selection

The SIMPUT catalog can cover an area on the sky larger than the Field of View (FOV) of the telescope. In order to save computation time, photons are only generated for sources inside the FOV. Sources outside this range are assumed to be invisible and therefore neglected. The applied selection criterion is the angular separation between the source position and the telescope axis, which is required to be less than the radius of the FOV.

3.2 Photon Time

The average photon rate R observed from a particular source is determined by its energy flux F_X in the reference band E_min to E_max, which is specified in the SIMPUT catalog, its photon spectrum $P(E)$, and the on-axis ARF (Ancillary Response File) of the instrument,

$$R = \frac{F_\mathrm{X}}{\int_{E_\mathrm{min}}^{E_\mathrm{max}} P(E)\,E\,\mathrm{d}E} \cdot \int_0^\infty P(E)\,\mathrm{ARF}(E)\,\mathrm{d}E \tag{1}$$

If a light curve is given by the SIMPUT file, it is piecewise linearly interpolated to create the photon rate for any time between the given values of the light curve. For more technical details and verification of the chosen algorithms, the interested reader may refer to Dauser et al. (2019).

3.3 Photon Energy

The energies of the photons produced for a particular source are distributed according to the spectrum obtained by multiplication of the instrument-independent flux density defined in the SIMPUT file, with the instrument-specific energy-dependent on-axis ARF (George et al., 1998, 2007). Since the spectrum and the ARF are defined on a particular energy grid and for discrete energy bins respectively, an interpolation of the spectrum to the energy bins of the ARF is applied. Drawing random energies with the inversion method (e.g., Deák, 1990; Gould et al., 2006) reproduces the selected spectral distribution.

3.4 Photon Direction of Origin

For a point-like source, the direction of the incident photons is equivalent to the location of the target on the sky. For a spatially extended source, a randomization of the photon direction is applied based on the image defined in the respective SIMPUT HDU. The relation between the image pixels k, l and the equatorial coordinate system used by SIXTE is defined by the World Coordinate System (WCS) header keywords (Greisen & Calabretta, 2002; Calabretta & Greisen, 2002) of the respective FITS image. The coordinate transformation is performed with the `WCSLIB`[13] library.

[13] `http://www.atnf.csiro.au/people/mcalabre/WCS/wcslib/`

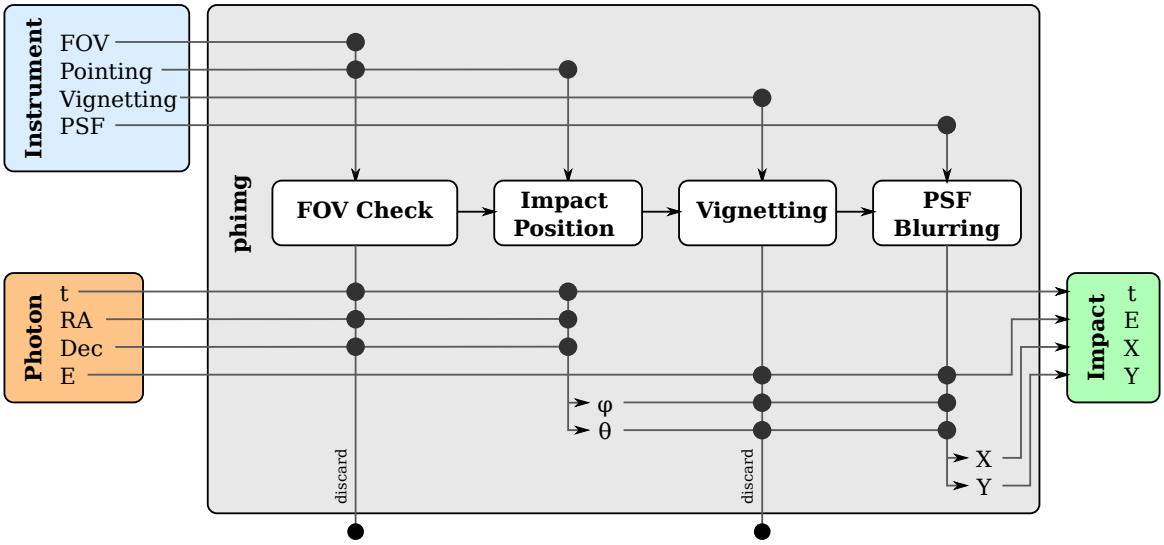

Figure 4: A block diagram of the photon imaging stage of SIXTE.

4 Photon imaging

Photons emitted by the simulated sources are imaged onto the detector plane by using calibration information describing the optical properties of the telescope. The input is a photon with the information about its arrival time t, its energy E and the direction of its origin, expressed in RA and Dec. When propagated through the optics, it is either discarded or converted into a photon impact with a time t, energy E, and a position (X, Y). Figure 4 shows the flow of information in the imaging stage of the simulation graphically. The individual steps are covered in the following sections.

4.1 PSF randomization

Focusing astronomical optics focus parallel rays of light to points on the focal plane. However, as the optics are not ideal, the image of a point source on the detector will be smeared. The smeared image is called the point spread function, or PSF. Most optics are designed to optimize the image and thus the shape of the PSF inside a certain region around the optical axis. With increasing off-axis angles, the PSF usually degrades to a complex shape which grows larger in size.

To account for non-ideal optics without performing actual ray tracing, SIXTE uses standard PSF calibration files to randomize the photon's impact position on the detector. The PSF is allowed to be dependent on the photon's Energy (E) and on the off-axis angle (θ) and the azimuthal angle (ϕ), i.e., is the angle between the intersection of the point $(X, Y) = (0, 0)$ and the ideal impact position and the instrument's X-axis.

For each individual photon, the PSF is interpolated linearly by randomly choosing one of the best matching entries in the calibration file. The resulting PSF is rotated accordingly to the difference between the simulated azimuthal angle and the one of the PSF chosen from the calibration file. This reduces the blurring of some asymmetrical features in complex PSF-shapes.

This PSF is interpreted as a spatial random distribution function for the final impact position of the photon.

4.2 Vignetting

Most imaging devices such as telescope mirrors are optimized for a certain region around the optical axis. With increasing off-axis angle, not only the PSF deteriorates, but the image also gets darker. This is commonly

described by the vignetting function, which is defined as the factor between the measured brightness of a source imaged at a given position and the measured brightness of the same source aligned to the optical axis.
SIXTE handles vignetting depending on the energy E of the photon, the off-axis angle θ and the azimuth angle ϕ (see Sect. 4.1 for the definition). The tabulated vignetting data is interpolated linearly for the energy and off-axis angle of each photon. Note that a dependence on the azimuthal angle is currently not yet implemented in Sixte. To take the vignetting into account in the simulation, a random number between 0 and 1 is drawn and compared to the resulting vignetting value, which is interpreted as the probability that the photon is actually imaged. If the random number is smaller than the vignetting value, it is imaged, if it is larger, it is discarded.

4.3 Instrument Coordinate System

The instrument coordinate system (X, Y) is defined as a planar, Cartesian system located in the focal plane behind the instrument's optics. Its units are meter, with the origin being the point of intersection between the optical axis and the focal plane. If an attitude file is used which has the keyword `ALIGNMEN` set to `MOTION`, the X-axis of the instrument coordinate system points along the direction of motion of the pointing vector. In any other case, the X-axis points towards the celestial north pole. The Y-axis then points towards the west. If additionally a Roll Angle is defined in the attitude file, the X- and Y-axes as described before are rotated around the optical axis by this angle.

5 Background Events

Background events in general must not behave like photons in terms of their origin or the interaction with the mirror system and detector. This is the reason why these events are generated between the imaging and detection steps inside SIXTE. Currently, there are in total two possibilities to insert background information into the simulation. Both contributions have to be specified accordingly in the XML file. The most simple way is to give the background as a PHA spectrum (Sect. 5.1). Additionally a background event list, including complete particle tracks, can be given (Sect. 5.2). For both contributions a light curve can be specified to make it time variable (Sect. 5.3)
In addition to the background contributions described in the following, X-ray background entering through the mirrors can always be described as an additional SIMPUT file in the simulation.

5.1 Sampling from a Background Spectrum

In the simple case, a spectrum is used to describe the background. This file needs to be a table with two columns, `CHANNEL` and `RATE`. The first column represents the RMF channel, to which the second column gives an event rate in units of cts s^{-1} m^{-2}. During the detection step, events are randomly sampled from this distribution, and treated like normal photon events thereafter. The XML element for this contribution is `phabackground` (see, e.g., the WFI example XML file at Sect. 6.1). Vignetting is not considered for these events by default, but can be applied if desired by setting the optional attribute `vignetting` within this element to the corresponding file.

5.2 Sampling from a Background Event List

Warning: This feature has not yet been ported to SIXTE version 3!
Additionally a background event list can be used in SIXTE to sample more complicated and realistic background events such as full particle tracks in the detector. Those events can for example be simulated with software like `Geant4` and need to be put in the proper format to be fed into SIXTE. The format is described in the `SIXTE-BKG` document. An excert is given below. Figure 5 shows an example of a WFI 50 sec simulation of the Tycho SNR with a very high particle background rate of 5 cm^{-2} s^{-1} (1000× exaggerated).
The XML tag for this contribution is `auxbackground` (see, e.g., the *eROSITA* XML file at Sect. 6.3). The input file is in the FITS format, required to be in the following format. Most importantly either the correct rate has to be given or the rate is calculated directly from the event list and the given detector dimensions.
Note that for the tools `sixtesim` and `gendetsim`, it is also possible to enable this background via the command line parameter `BkgEventList`. In this case, the file must contain a `RATE` header key.

5.2.1 Background Event List Format

primaryid: Identification number of the primary particle causing one or multiple events on the detector
edep: Deposited energy at the current position (X, Y, and Z) *Recommended unit: keV*
time: *optional* time when the energy is deposited *Recommended unit: s*
X: x-position in detector coordinates (not pixel) *Recommended unit: mm*
Y: y-position in detector coordinates (not pixel) *Recommended unit: mm*
Z: *optional* depth of the interaction *Recommended unit: mm*

5.2.2 Header Keywords

EVTSORT *mandatory* All particle IDs consecutively ordered and within each primary ID sorted by time.
TUNITx *mandatory* Units of the columns have to be given.

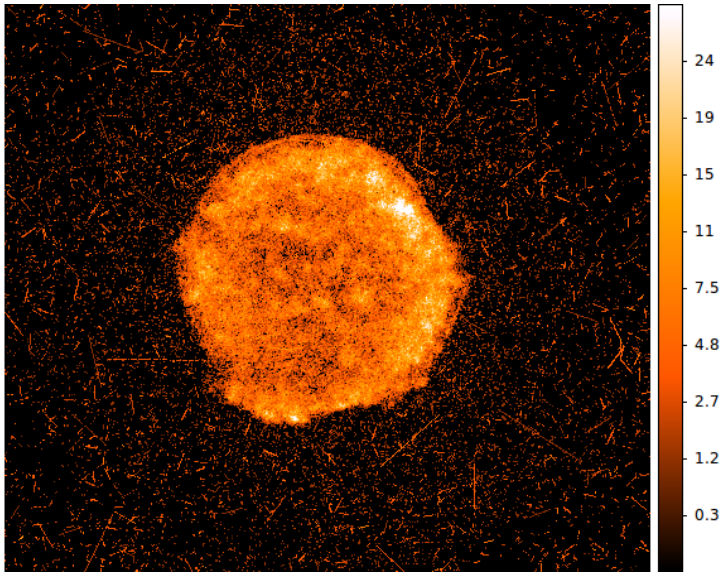

Figure 5: An example of a simulation with a background event file showing simulated particle tracks and the Tych SNR. It is simulated for 50 sec, with a very high particle background rate of 5 cm^{-2} s^{-1}, exaggerated by a factor of 1000.

RATE *optional* background event rate on the complete device (particles s^{-1} cm^{-2}); can be overwritten by the simulation setup (in the instrument XML)
PHOTONS *optional* boolean key deciding whether or not the background is simulated as a photon background, meaning charge clouds are simulated. If the keyword does not exist, it is assumed to be false
ORIGIN institute producing the file
DX size in x-direction of the volume where the energy is deposited
DY size in y-direction of the volume where the energy is deposited
DZ size in z-direction of the volume where the energy is deposited
POSDEL the volume in which the energy is deposited is defined by DX, DY, DZ, and its position X, Y, Z. The POSDEL keyword (respectively the POSDELX, POSDELY, POSDELZ keywords) define the location of the point X, Y, Z, with respect to this volume. Either POSDEL, or all three of POSDELX, POSDELY, POSDELZ must be given. Values range from 0 (left border) to 1 (right border).
POSDELx *see POSDEL*
POSDELy *see POSDEL*
POSDELz *see POSDEL*
SAMINX min X of sensitive detector area [mm]
SAMAXX max X of sensitive detector area [mm]
SAMINY min Y of sensitive detector area [mm]
SAMAXY max Y of sensitive detector area [mm]
INSTRUMENT instrument used for the simulation
DETECTOR detector used for the simulation

5.3 Time Variable Background in SIXTE

Warning: This feature has not yet been ported to SIXTE version 3!

Time variable background can be given as SIMPUT standard[14] light curve together with PHA or Event List background. It is defined such that a value of 1 in the light curve corresponds to the rate specified by the given background file.

```
<auxbackground filename="bkg_evts.fits" lightcurve="bkg_lc.fits"/>
```

[14] https://www.sternwarte.uni-erlangen.de/sixte/sources/

Table 1: Chip and read-out data of the simulated DEPFET setups. The read-out time per line is set to 9.8 μs for the LDs and 2.5 μs for the FD. The FD is assumed to be defocused by default. Most configurations are given without (`wo`) and with (`w`) optical blocking filter. These two different options are provided in separate directories (`wfi_wo_filter_B4C` and `wfi_w_filter_B4C`). Note that the *full* mode XML contains four chips which are simulated all at once (see Sect. 10.4).

Name	**Filename**	**Size** (rows × columns)	**time resolution**	**defocusing**	**filter**	**HEW**
full	`ld_wfi_ff_all_chips.xml`	(4×) 512 × 512	5018 μs	—	wo/w	5″
indiv	`ld_wfi_ff_chip[0,1,2,3].xml`	512 × 512	5018 μs	—	wo/w	5″
large	`ld_wfi_ff_large.xml`	512 × 512	5018 μs	—	wo/w	5″
w128	`ld_wfi_w128.xml`	128 × 512	1254 μs	—	wo/w	5″
w256	`ld_wfi_w256.xml`	256 × 512	2509 μs	—	wo/w	5″
fast	`fd_wfi_df35mm.xml`	64 × 64	80 μs	35 mm	w	
fastBe	`fd_wfi_df35mm_Be100.xml`	64 × 64	80 μs	35 mm	w	

6 Specific Instruments

In this section a selection of instruments is presented for which SIXTE simulations can be performed. Usually the instrument description and additional tools are available for these instruments.

6.1 *Athena* – WFI

The *Athena*-WFI is a silicon-based DEPFET (DEPleted Field Effect Transistor)-detector currently planned for the *Athena* mission.

6.1.1 Chip Geometries and Readout Modes

In Table 1, the chip geometries and readout modes which are available for the WFI for download[15] are listed. Each pixel has an edge length of 130 μm. The charge cloud is modeled as a Gaussian distribution with σ = 11 μm (see Appendix D for the definition). There are options for the Large Detector Array (LDA) of the WFI with all four Large Detector (LD) chips (`full`), a full frame mode of a single LD chip (`large`), and two window modes (*w128*, *w256*), where only part of the chip is read out in order to mitigate pile-up. The WFI chips are modeled in the DEPFET framework (see Sect. 7.2) with a read-out time per line of 9.8 μs. Moreover, readout modes for the separate Fast Detector (FD) with an optical blocking filter (`fast`) and with a thick 100 μm Be filter (`fastBe`) are available. The FD is defocused by 35 mm in the current baseline concept and will be read out almost a factor four faster than the LDA (2.5 μm per line). In addition, the FD chip is split in two hemispheres which are read out simultaneously, further increasing the read-out speed by a factor of two.
All of these configurations and the assumptions entering each are described in more detail in the WFI configuration document (ECAP-WFI-CONF-20211116).

6.1.2 WFI Example XML File

```
<?xml version="1.0"?>
<instrument telescop="Athena" instrume="WFI">
<telescope>
<arf filename="athena_wfi_sixte_13rows_wo_filter_LDA_v20240209.arf"/>
<focallength value="12.0"/>
<fov diameter="1.0"/>
```

[15] `https://www.sternwarte.uni-erlangen.de/sixte/instruments/`

```
<psf filename="athena_psf_hew9_20231211.fits"/>
<vignetting filename="athena_vig_13rows_20231211.fits"/>
<pha2pi filename="athena_wfi_pha2pi_v20230609.fits"/>
<pirmf filename="athena_wfi_pirmf_v20230609.rmf"/>
</telescope>

<detector type="depfet">
<dimensions xwidth="512" ywidth="512"/>
<wcs xrpix="256.5" yrpix="256.5" xrval="0.0" yrval="0.0" xdelt="130.e-6" ydelt="130.e-6"/>
<depfet integration="1.0e-6" clear="1.0e-6" settling_1="5e-7" settling_2="4e-7" type="normal"/>
<rmf filename="athena_wfi_sixte_v20230523.rmf"/>
<phabackground filename="sixte_wfi_particle_bkg_20230602_nxb8.pha"/>
<split type="GAUSS" par1="11.e-6"/>

<threshold_readout_lo_keV value="60.e-3"/>
<threshold_event_lo_keV value="60.e-3"/>
<threshold_split_lo_keV value="60.e-3"/>
<readout mode="time">
  <loop start="0" end="511" increment="1" variable="$i">
    <wait time="3.9e-6"/>
    <readoutline lineindex="$i" readoutindex="$i"/>
  </loop>
  <newframe/>
</readout>
</detector>
</instrument>
```

6.2 *Athena* – X-IFU

6.2.1 Baseline focal plane geometry and simulation parameters

X-IFU and other microcalorimeters require additional information relevant to their specific readout scheme, as seen in the `readout` tag of the following XML:

```
<?xml version="1.0"?>
<instrument telescop="Athena" instrume="XIFU">
  <telescope>
    <focallength value="12.0"/>
    <fov diameter="0.5"/>
    <arf filename="instdata/athena_xifu_13_rows_no_filter.arf"/>
    <psf filename="instdata/athena_psf_hew9_20240326.fits"/>
    <vignetting filename="instdata/athena_vig_13rows_20240326.fits"/>
  </telescope>

  <detector type="microcal">
    <rmf filename="instdata/athena_xifu_4eV_gaussian.rmf"/>
    <phabackground filename="instdata/athena_xifu_nxb_sixte.pha"/>
    <threshold_readout_lo_keV value="0.2"/>
    <threshold_event_lo_keV value="0.2"/>
```

```
    <geometry type="free" npix="1504" xoff="158.5e-6" yoff="158.5e-6" >
      <hexagonloop radius="0.00760" pixelpitch="0.000317" cross="1">
          <pixel>
              <shape posx="$x" delx="$p" posy="$y" dely="$p" width="310.7e-6" height="310.7e-6"/>
          </pixel>
      </hexagonloop>
    </geometry>

    <readout mode="event" samplefreq="130.2083e+3"/>

    <reconstruction>

    <grading num="1" name="veryhigh" pre="3125" post="7192" rmf="instdata/athena_xifu_4eV_gaussian.rmf"
    <grading num="2" name="high" pre="3125" post="3496" rmf="instdata/athena_xifu_4eV_gaussian.rmf"/>
    <grading num="3" name="int" pre="1563" post="1448" rmf="instdata/athena_xifu_4d2eV_gaussian.rmf"/>
    <grading num="4" name="mid" pre="1563" post="412" rmf="instdata/athena_xifu_5eV_gaussian.rmf"/>
    <grading num="5" name="lim" pre="1563" post="156" rmf="instdata/athena_xifu_7eV_gaussian.rmf"/>
    <grading num="6" name="low" pre="1563" post="7" rmf="instdata/athena_xifu_30eV_gaussian.rmf"/>

    <crosstalk>
      <mux type="tdm" channel_freq_list="instdata/lpa_tdm_mux47_317um_random_20240530.dat"/>
      <propcrosstalk filename="instdata/proportional_xt_LPA50x30_20240530.fits"
        scaling1="2e-2" scaling2="1e-4" scaling3="7e-4" trig_thresh="0.2"/>

      <dercrosstalk filename="instdata/mut_ind_xt_LPA50x30_20240530.fits"
        scaling="1" trig_thresh="1e12"/>

      <thermalcrosstalk>
        <pair distance="317.01e-6" weight="3.402e-4"
          timedepfile="instdata/thermal_xt_nearest_time_dep_20240614_sixtesim.fits"
          trig_thresh="10"/>
        <pair distance="448.31e-6" weight="2.334e-4"
          timedepfile="instdata/thermal_xt_diagonal_time_dep_20240614_sixtesim.fits"
          trig_thresh="1e12"/>
        <pair distance="634.01e-6" weight="1.051e-4"
          timedepfile="instdata/thermal_xt_second_time_dep_20240614_sixtesim.fits"
          trig_thresh="1e12"/>
      </thermalcrosstalk>
    </crosstalk>
    </reconstruction>

  </detector>
</instrument>
```

This baseline simulation configuration comprises 1504 square pixels of size 310.7 µm separated by 6.3 µm gaps on a hexagonal grid.
Four different grades are currently defined, ranging from grade 1 to 6: very high and high resolution (4 eV), intermediate resolution (4.2 eV), medium resolution (5 eV), limited resolution (7 eV) and low resolution (30 eV), all resolutions degrading linearly after 7 keV.
Additionally, there are two more cases that may occur in simulations:

- Pile-up. Pulses are separated by less than a single sample time and are merged into one event.
- Invalid events: the generated photon is outside the energy band 0.2 - 12 keV (this happens since the energy band of the ARF sometimes goes beyond 12 keV) or two events are closer in time to each other than the

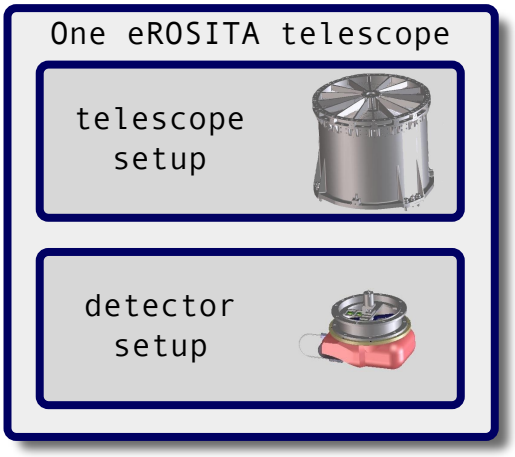

Figure 6: Schematic setup of one of the seven eROSITA sub-instruments. The corresponding XML configuration file specifies the mirror module and the detector module with its respective parameters.

worst grade allows for. In both cases, events are marked by the grade -1.
Lastly, the effect of crosstalk is modeled by a series of lookup tables and tags describing the different crosstalk mechanisms, further described in Appendix E.

6.3 eROSITA

The XML code below represents the definition of one of the seven eROSITA sub-instruments consisting of a telescope and a detector, as illustrated by the sketch in Fig. 6.

```
<?xml version="1.0"?>
<!-- SRG eROSITA version 1.4 -->
<instrument telescop="eROSITA" instrume="FM1">

<telescope>
  <!-- Telescope 3: onchip filter, open filter -->
  <arf filename="arf01_200nmAl_sdtq.fits"/>
  <fov diameter="1.02"/>
  <focallength value="1.6"/>
  <vignetting filename="erosita_vignetting_v2.1.fits"/>
  <psf filename="erosita_psf_v3.1.fits"/>
</telescope>

<detector>
  <dimensions xwidth="384" ywidth="384"/>
  <pixelborder x="0." y="0."/>
  <wcs xrpix="192.5" yrpix="192.5" xrval="0." yrval="0."
       xdelt="75.e-6" ydelt="75.e-6" rota="270"/>
  <rmf filename="sixte_erormf_normalized_singles_20170725.rmf"/>
  <cte value="1"/>
  <split type="exponential" par1="0.355"/>
  <auxbackground filename="merged_hitlist_converted.fits"/>
  <threshold_readout_lo_keV value="0."/>
  <threshold_event_lo_keV value="0.0"/>
  <threshold_split_lo_fraction value="0.01"/>
```

```
  <threshold_pattern_up_keV value="10."/>

  <readout mode="time">
    <wait time="0.05"/>

    <loop start="0" end="383" increment="1" variable="$i">
      <readoutline lineindex="0" readoutindex="$i"/>
      <lineshift/>
      <wait time="0.3e-6"/>
    </loop>

    <newframe/>
  </readout>
</detector>

</instrument>
```

The telescope has a FOV with a diameter of 1.02 deg and a focal length of 1.6 m. The PSF and vignetting function are described in two separate files including several data sets for different energies and off-axis angles. The detector consists of 384×384 pixels with an area of $75\,\mu\text{m} \times 75\,\mu\text{m}$ each. The applied coordinate system is centered on the detector. Its origin coincides with the intersection of the optical axis of the telescope with the detector plane. The pixels have no borders. The presented detector model does not include a particular framestore area as the real eROSITA Charge Coupled Devices (CCDs) (Meidinger et al., 2006, 2007, 2008, 2009, 2010, 2011). Instead the collected charges in the sensitive part of the pixel array are read out with the same speed as the transfer time to the framestore area would require. This approach is, of course, only possible in the simulation and avoids the necessity of defining a framestore region.
The combined ARF of the instrument, which takes into account the effective area of the telescope and the quantum efficiency of the detector, is stored in the file `erosita_iv_1telonaxis_ff.arf`. The energy resolution of the detector and related effects such as an escape peak are described by the RMF. A Charge Transfer Efficiency (CTE) of 100 % is assumed and charge cloud splitting between neighboring pixels is simulated according to the model developed by K. Dennerl (priv. comm.). Random background events are inserted from the data set in the file `merged_hitlist.fits`.
The lower threshold for the recorded events is set to 0 keV, and the upper threshold to 12 keV. The pattern analysis algorithm selects only events with a signal of more than 150 eV as main events. Split partners are searched with a lower threshold of 1 % of the total signal around the location of the main event.
The readout mode of the CCD is characterized by an exposure time of 50 ms followed by 384 subsequent readouts and line shifts, which is illustrated in Fig. 7. As mentioned above, instead of using a framestore area, for the readout process a high line shift rate is used. A single line shift lasts 0.3 μs, i.e., the readout of the whole array takes about 115 μs and is equivalent to the transfer from the image to the framestore area.
With this XML description the essential parameters of a single eROSITA telescope and detector sub-system are defined. Similar XML definition files for various instruments can be found in the SIXTE software package.

6.3.1 Specific Tools

The following tools were developed for eROSITA.

- `ero_vis`
  Determines the visibility of X-ray sources at particular positions in the sky during the eROSITA all-

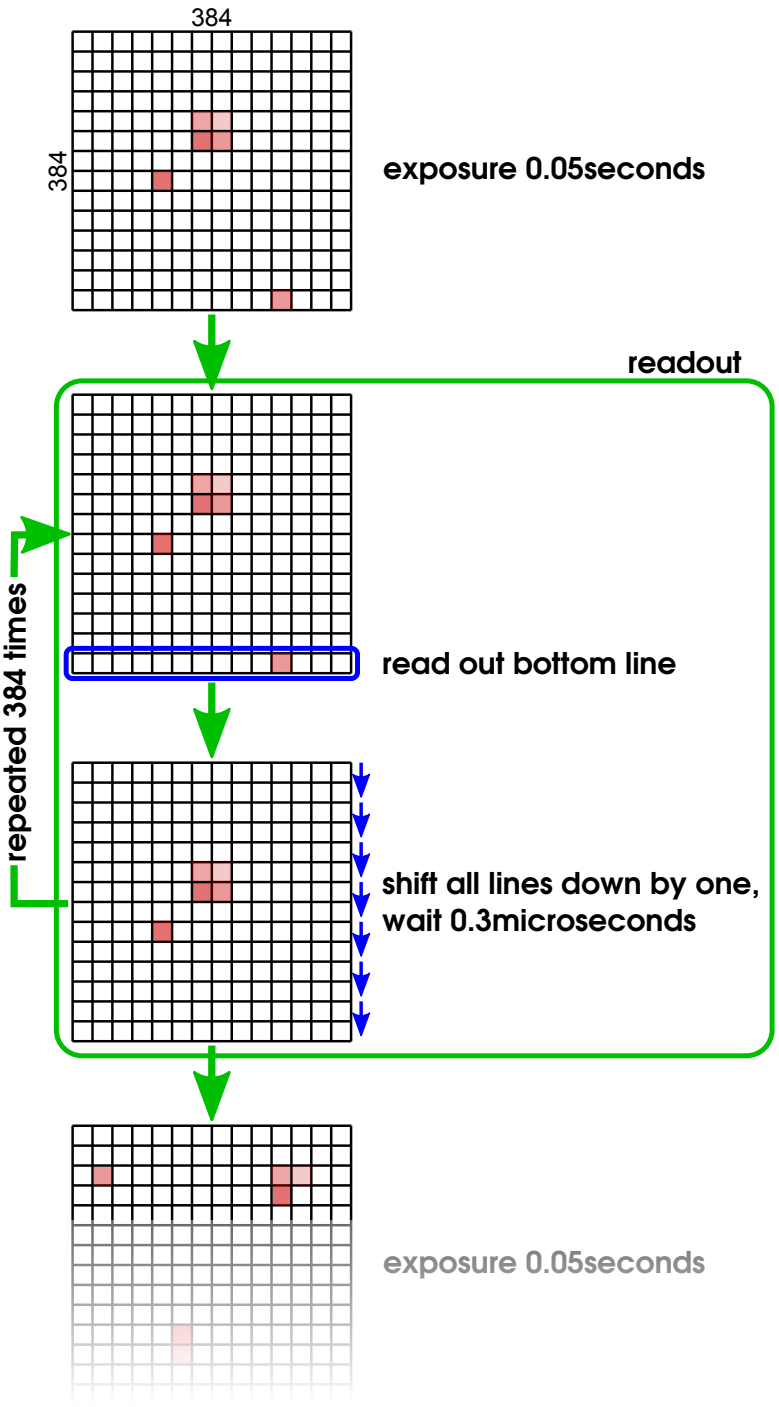

Figure 7: Schematic implementation of the readout mode of an eROSITA CCD. For illustrative purpose not all 384×384 pixels are drawn. Each exposure time of 50 ms is followed by 384 subsequent readouts and line shifts. As indicated in the text, in the simulation the transfer of signals to the framestore area, which is typical for the eROSITA CCDs, is replaced by a faster readout with an equivalent time interval.

sky survey based on the corresponding attitude file and produces a GTI file, which can be input in the simulation to simulate only when sources are in the field of view.

- `ero_rawevents`
  Converts a generic event list to the eROSITA-specific raw event file format, which resembles the output of the instrument before processing by data analysis software.
- `ero_calevents`
  Converts a generic event list to the eROSITA-specific calibrated event file format. In comparison to the

raw data, it also contains information, e.g., about split patterns or associated sky positions and resembles the output of the Standard Analysis Software System (SASS).

6.4 *XMM-Newton* – EPIC-pn

Another example for the capabilities of the introduced XML format is illustrated by the code below, which represents the definition of the burst mode (Kendziorra et al., 1997, 1998, 1999; Kuster et al., 1999) of the European Photon Imaging Camera (EPIC) pn camera (Strüder et al., 2001) aboard *XMM-Newton* (Jansen et al., 2001). This mode is designed for observations of very bright X-ray sources. As the setup of an instrument configuration has already been shown in 6.3, only the XML tags for the definition of the readout mode are listed here.

```
<readout mode="time">
  <loop start="0" end="199" increment="1">
    <clearline lineindex="0"/>
    <lineshift/>
    <wait time="0.72e-6"/>
  </loop>
  <loop start="0" end="179" increment="1"
        variable="$i">
    <readoutline lineindex="0"
                 readoutindex="$i"/>
    <newframe/>
    <lineshift/>
    <wait time="23.04e-6"/>
  </loop>
</readout>
```

First 200 fast line shifts are performed such that each row is shortly exposed to the observed target, which has to be located close to line number 190. Then the collected signals are read out requiring 180 slower normal line operations (Kuster et al., 1999). The signal collected in the remaining upper 20 lines is not used.
A schematic illustration of this particular operation mode is shown in Fig. 8. It avoids pile-up at the cost of losing spatial information along the y-direction. A detailed description can be found in the previously cited references.

6.5 *Suzaku*

Similar to the EPIC-pn, the X-Ray Imaging Spectrometer (XIS) aboard *Suzaku* in the normal mode provides a burst option in order to avoid pile-up for observations of bright X-ray sources. With this option, the exposure time b can be set to an arbitrary value in 1/32 s steps (Koyama et al., 2007).
Before the actual exposure takes place, the detector waits for an interval of 8 s − b. Then all signals collected in the 1024×1024 pixels of the CCD during this period, are transferred out of the imaging area and flushed without recording. In parallel charge is injected into particular rows of the CCD according to the spaced-row charge injection mechanism, which mitigates the effects of radiation damage (Bautz et al., 2004; Koyama et al., 2007). The entire procedure results in a transfer time of 156 ms.
The signals collected during the subsequent science exposure of length b are transferred to the framestore area and read out from there. As this second transfer takes place without charge injection, it only requires 25 ms.
The described detector operation can be implemented in SIXTE with the appropriate XML tags, e.g. for an exposure $b = 0.094$ s:

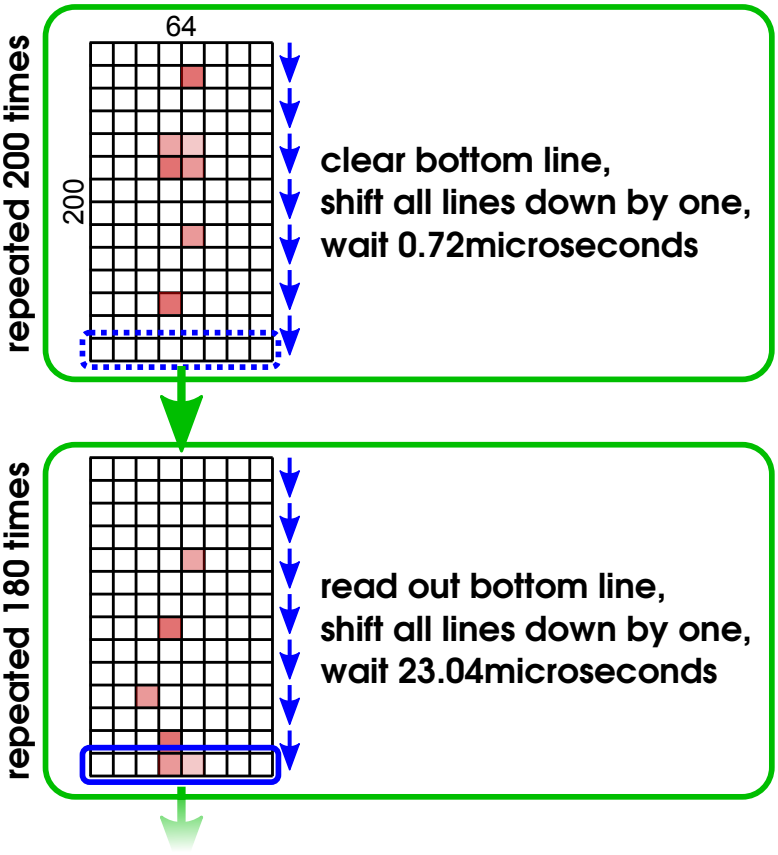

Figure 8: Schematic implementation of the burst readout mode of the EPIC pn camera aboard *XMM-Newton*. For illustrative purpose not all 64×200 pixels are drawn. After 200 fast lines shifts, the detector is read out with 180 subsequent normal line operations.

```
<readout mode="time">
  <wait time="7.906"/>
  <loop start="0" end="1023"
        increment="1">
    <clearline lineindex="0"/>
    <lineshift/>
    <wait time="0.000152"/>
  </loop>
  <wait time="0.094"/>
  <loop start="0" end="1023"
        increment="1" variable="$i">
    <readoutline lineindex="0"
                 readoutindex="$i"/>
    <lineshift/>
    <wait time="0.0000244"/>
  </loop>
  <newframe/>
</readout>
```

Analogous to the model of the eROSITA CCD introduced in 6.3, the framestore area, which is part of the real device, is not implemented in the detector model, but the charges are directly read out from the bottom line of the CCD. Due to the different duration of the first and the second charge transfer process with and without

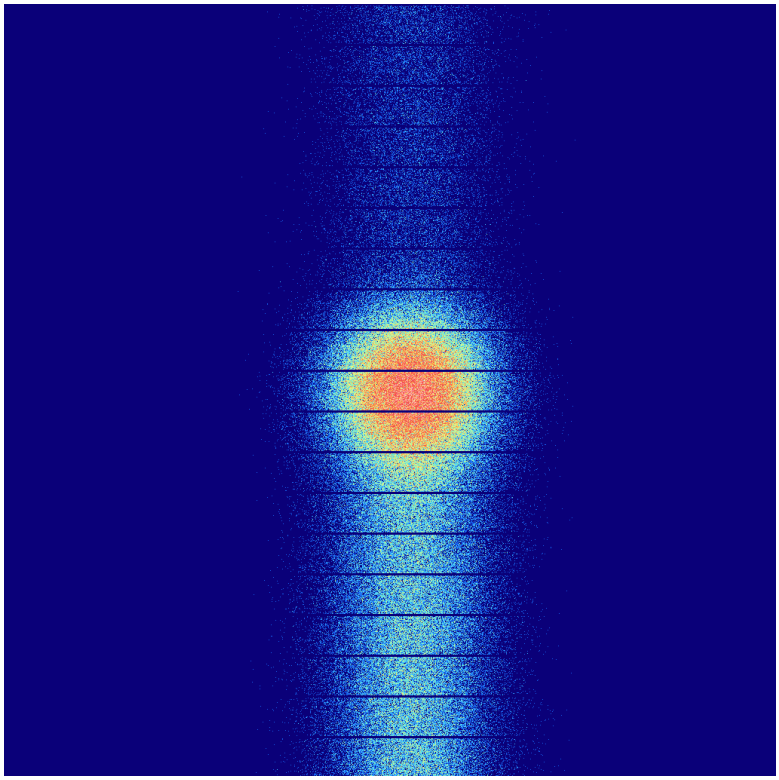

Figure 9: Simulated observation of the Crab nebula with one of the *Suzaku* XISs (chip width corresponds to 17′) using the burst option. The readout direction is towards the bottom and the image is given in linear scaling. The evident asymmetric density of OOT events above and below the source is caused by the different speed of the two charge transfer processes before and after the science exposure of $b = 0.094$ s.

charge injection, the images obtained from observations in burst mode exhibit an asymmetric distribution of Out Of Time (OOT) events[16], as illustrated in Fig. 9.

[16] http://www.astro.isas.jaxa.jp/suzaku/doc/suzaku_td/

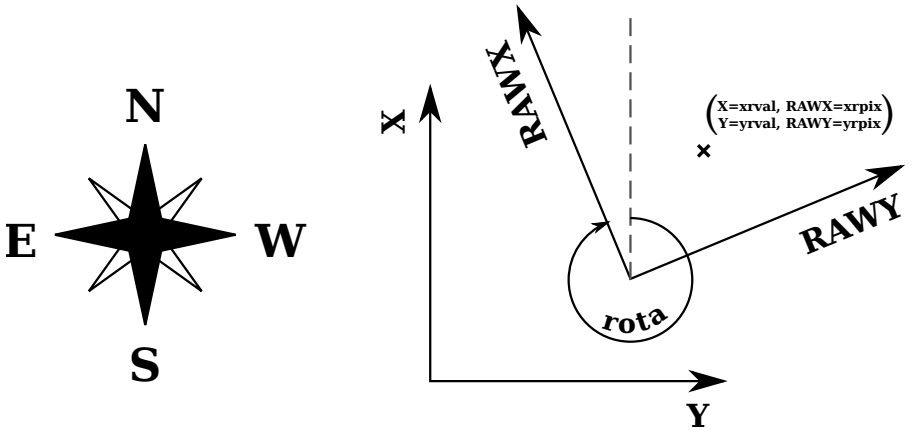

Figure 10: The coordinate systems as used by SIXTE, seen as projected onto the sky. In this figure, the X-axis points towards north. It can be realigned using an attitude file. The RAWX and RAWY axes can be rotated by the `rota`-key in the detector XML description. The angle is measured from the *X*- to the RAWX-axis clockwise when projected onto the sky.

7 Photon Detection for Silicon Detectors

7.1 Camera Coordinate System

As described in Sect. 4.3, the photon impact positions are kept in a Cartesian *X*-, *Y*- coordinate system, which is aligned either to the north pole or to a direction defined in the attitude file. This coordinate system is used to describe the focal plane of the instrument.
To describe a detector, another coordinate system is introduced. Its new axes are named RAWX and RAWY and are measured in units of pixels. The size of the pixels is defined via the keys `xdelt` and `ydelt`. The coordinate system can be shifted with respect to the instrument system. This is done via the XML-detector description and its WCS coordinate tags. The keys `xrval` and `yrval` define a point in the X,Y-plane. The keys `xrpix` and `yrpix` define the respective pixel coordinates which are associated with this point. The new system can also be rotated via the `rota`-key, using the angle from the X to the RAWX-axis, measured clockwise when projected onto the sky. Fig. 10 illustrates the coordinate systems as used by SIXTE.
The detector dimensions (i.e., number of pixels) are specified by `xwidth` and `ywidth`. Note that in silicon-type detectors photons that hit pixels at the edge are discarded, the reason being that in this case eventual split partners might be lost and therefore a significant fraction of these events will have a wrong energy assigned. Therefore the sensitive area of the detector is [`xwidth`−2]×[`ywidth`−2].

7.2 DEPFET Specific Features

The previous chapters described a generic detector implementation which has pixels ordered in lines and columns. This principle is also valid for the special case of DEPFET detectors, where each pixel can be addressed and read out on its own. Due to the architecture of the read-out electronics, these detectors are often read out line per line, too, which allows to use the same functionality.
However, DEPFETs show a more complex behaviour during read-out which is implemented in SIXTE. The next sections give an overview about the simulation relevant principles of DEPFETs and how they affect the simulated measurements.

7.2.1 DEPFET principles

A DEPFET detector consists of pixels which can be addressed individually. In contrast to a CCD, the charge produced by a photon does not need to be shifted over other sensitive parts of the detector to reach the readout electronics. Instead, for the purpose of the simulation, the charge held in a pixel can be regarded as read out just at the point where it was created.
This read out process however is not an instantaneous task but consists of several phases. In the case of a standard DEPFET, the read-out sequence for each line starts with a settling time, followed by a first integration time, where the collected charge is integrated. After this first measurement, the charge is removed during the clear interval. Then, a second settling time is followed by the second integration, where the baseline is measured. By subtracting this measurement from the first one, an equivalent of the photon's signal can be estimated. To read out the whole detector, all lines are read out one after each other, or several lines at the same time. All other lines are in the exposure mode. In the case of a gateable DEPFET, the charge produced by a photon is kept in a storage region in each pixel during the exposure time, unaccessable for the read-out electronics. Instead, in a short interval at the beginning of one frame, all pixels transfer the collected charge to the read-out region of the pixels. Then, the read-out of the lines follows the same scheme as for the standard DEPFET.
This read-out mechanism influences the quality of the measurements. Events which hit the detector during the read-out or transfer intervals can not be measured reliably but cause so-called "misfits". In these cases, the charge produced by the photon is only partially measured, and might be also carried over into the next frame. For standard DEPFETs, this process is described by three time parameters, the settling, integration, and clear times. In a linear approximation of the measurement processes, these three parameters explicitly define the measured photon energy by knowing the real photon energy and the impact time. The gateable DEPFET has an additional parameter, the transfer time. In a simplified scheme, it describes the time interval in which the storage is connected to the read-out circuit.

7.2.2 DEPFET implementation

The DEPFET read-out is not an instantaneous determination of the photon's energy but it integrates first the photon's voltage signal together with the signal baseline, and after the photon's charge is removed by the clear, the baseline is integrated again to remove it from the measurement. In a simplified scheme, the measurement can be described with three time intervals and the corresponding integration factors, as depicted in Fig. 11.
After the initial settling, the read-out of a standard DEPFET begins at t_0. Between t_0 and t_1, the signal present at the internal gate is integrated. Between t_1 and t_2, any charge is removed from the DEPFET's internal gate. Between t_2 and t_3, the second settling occurs. Between t_3 and t_4, the signal is integrated again but with negative sign. The result is, that the baseline signal can be substracted from the first measurement, providing an estimate of the photon's signal.
Misfits are photons which hit the detector during the integration time intervals. As their signal is not integrated for the full integration time, they can produce wrong measurements. The resulting measurement E_m of their true energy E_p can be described dependent on their impact time t_p:

- $t_p < t_0$ or $t_p > t_4$: $E_m = E_p$
- $t_0 < t_p < t_1$: $E_m = E_p \times (t_1 - t_p)/(t_1 - t_0)$
- $t_2 < t_p < t_3$: $E_m = -E_p$
- $t_3 < t_p < t_4$: $E_m = -E_p \times (t_4 - t_p)/(t_4 - t_3)$

In the case of $t_3 < t_p < t_4$, the charge is not removed in this read-out cycle and will be measured again during the next cycle, before it is cleared.

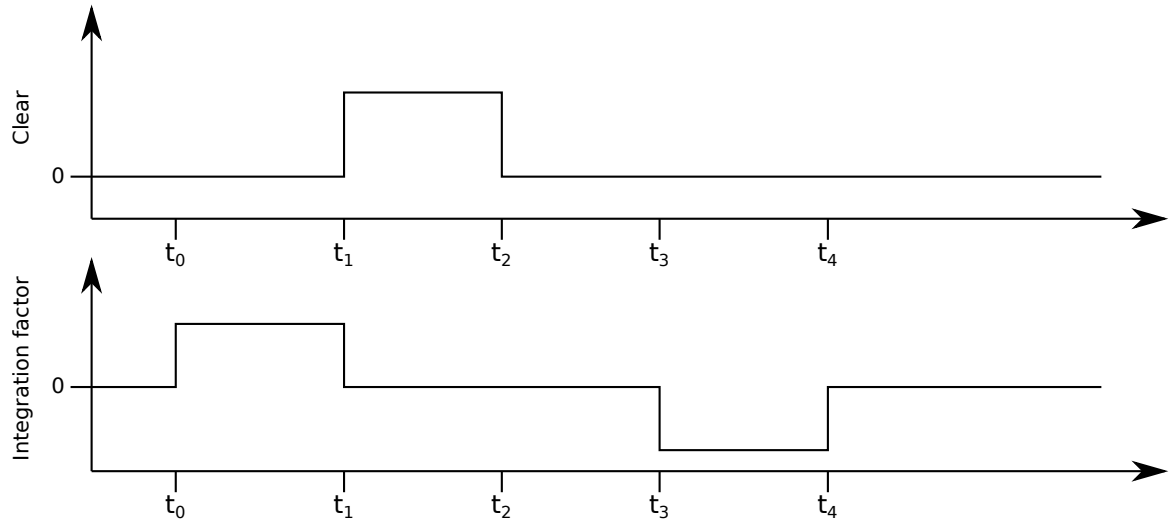

Figure 11: A basic scheme for the DEPFET read-out.

Another effect to be simulated is the limited clear speed. Instead of removing the charge instantaneously, it is cleared linearly between t_1 and t_2. If a photon hits the detector during this interval and causes a charge q_p, it is only partially cleared and leaves a remaining charge q_r, according to

$$q_r = q_p \times (t_p - t_1)/(t_2 - t_1) \ .$$

The remaining charge will be measured in the second integration time and causes a negative energy measurement of the value

$$E_m = -E_p \times (t_p - t_1)/(t_2 - t_1) \ .$$

The charge will be measured again in the next read-out cycle. If the impact is between t_2 and t_3, the charge is remaining completely. If two photons impact in the same pixel during one read-out cycle, their signals add to each other.
In case of the gateable DEPFET, a photon which arrives during this time interval is registered with reduced energy, as the charge can only be transferred partially. The remaining charge is transferred in the next cycle. For the gateable DEPFET, the clear is working in a similar way as before. Depending on the arrival time of the photon during this time interval, the resulting charge is removed partially. The only difference is that no negative charge in the current frame is produced, as the read out mechanism is shielded. The integration and settling time lengths do not influence the energy measurement of any photons in the gateable DEPFET.
In summary, one photon can produce up to two measured events, depending on the arrival time. Tables 2 and 3 list the measured energies for the pimary and secondary event, where primary means the measurement in the old frame and secondary meaning the event in the new frame.

Table 2: Possible detected events and their energy for the standard DEPFET, for an incoming photon of energy E_p. Marked in bold are only the events, which will result in an energy different from the input energy. The current stage of the read-out takes place from t_0 until t_1, and t_p denotes the arrival time of the photon. Note that these are simplified linear models (taken from the DRT report).

non-gateable	primary	secondary
1st settling	E_p	0
1st integ	$\boldsymbol{E_p(t_{1,\mathrm{integ}} - t_p)/\Delta t}$	0
clear	$\boldsymbol{E_p(t_p - t_{0,\mathrm{clear}})/\Delta t}$	
2nd settling	< 0	E_p
2nd integ	< 0	E_p

Table 3: Possible detected events and their energy for the gateable DEPFET, similar to Tab. 2. Marked in bold are only the events, which will result in an energy different from the input energy. Again the current stage of the read-out takes place from t_0 until t_1, and t_p denotes the arrival time of the photon.

gateable	primary	secondary
transfer (global)	$\boldsymbol{E_p(t_p - t_{0,\mathrm{transfer}})/\Delta t}$	$\boldsymbol{E_p(t_{1,\mathrm{transfer}} - t_p)/\Delta t}$
1st settling	E_p	0
1st integ	E_p	0
clear	0	$\boldsymbol{E_p(t_p - t_{0,\mathrm{clear}})/\Delta t}$
2nd settling	E_p	0
2nd integ	E_p	0

8 Photon Detection for Calorimeter Detectors

In the case of calorimeters, the approach used for silicon detectors to describe and simulate the detection plane based on lines and columns is not well adapted. A specific detector setup has therefore been developed in which each pixel is described individually (see Sect. 8.2). As of today two simulation processes using this description exist: one based on the simple randomization of the energy of the impacting photons (see Sect. 8.2) and the other on the actual simulation of the full read-out circuit for Transition Edge Sensor (TES)-based instruments (see Sect. 8.3).

8.1 Readout Description

The readout of microcalorimeters differs from that of silicon detectors, and requires additional information.

8.1.1 Grading

In a calorimeter, the impacting photons create signal pulses whose magnitude and shape depend on their energy. These pulses are then usually reconstructed using signal processing techniques either by a ground segment equipment or an on-board processor. These techniques typically have a degraded performance when pulses are close-by in time.
To represent this property, a grading scheme is defined with which each detected photon gets affected a grade and a corresponding RMF depending on the time intervals since the previous event and until the next one in the pixel timeline. In the SIXTE XMLs, grading rules are specified via the tag
`<grading num="..." name="..." pre="..." post="..." rmf="..."/>`
, where an event separated by more than `pre` samples from the previous event and `post` samples from the next one will be affected the grade `num` and its energy will be randomized using the RMF matrix `rmf`. `name` is just a tag to ease reading.
These grading rules are tested in a top-down exclusive way using the order specified in the XML: each event gets the grade corresponding to the first rule it is compliant with. Furthermore, events separated by less than one readout sampling interval are considered to pileup and become a single event with the sum of the initial energies that is then treated as a normal event.

8.1.2 Crosstalk

This detector type also offers the possibility to simulate crosstalk between pixels caused by thermal diffusion or the potential multiplexing scheme. It is currently modeled using a linearized approach with “coupling matrices” and higher-dimensional lookup tables for nonlinear effects. More details can be found in Appendix E.

8.2 RMF based simulation

The first simulation type available for calorimeters is a fast tool modeling the major scientific capabilities of the instrument. It uses an abstraction of the detection process based on several energy redistribution matrices and is typically suited for full end-to-end simulations, especially those for which the full imaging capability of the instrument is required (see Sect. 6.2 for more details on the specific X-IFU setup).
In this simulation approach, the grading of individual events is determined using the rules outlined in Sect. /ref-sec:grading. After the grade is identified, the read-out energy of each event is randomly selected using the RMF matrix corresponding to its grade.
Furthermore, crosstalk as described in Appendix E is applied using an approach based on pixel-to-pixel coupling matrices and lookup tables to model the energy shift caused by crosstalk in coupled pixels.

8.3 TES simulations

SIXTE also incorporates a slower, but more representative device-level simulator for TES based instruments which is an ab initio representation of the physics of the detection process. It consists of two subtools: `tessim` simulates the read-out signal of a TES pixel following X-ray impacts and `SIRENA` is an implementation of the optimal filter algorithm that estimates the photon energies from the readout signal. Even though both tools are incorporated in the SIXTE framework, they are currently not integrated in the generic end-to-end simulations pipeline such that they can only be used starting from the `piximpact` file stage. This simulation branch is typically suited for the detailed study of instrumental effects such as energy resolution degradation with energy or count rate and is often used to provide the energy redistribution and coupling matrices used by the RMF based simulator.
In general, this tool is intended to be used by specialists only. For virtually all science simulations for calorimeters, the RMF based tools should be used.

8.3.1 tessim

`tessim` is an improved and extended version of a TES simulation code originally developed by Stephen J. Smith (NASA GSFC). In brief, the code performs a numerical solution of the differential equations for the time-dependent temperature, $T(t)$, and current, $I(t)$, in the TES (Irwin & Hilton, 2005),

$$C\frac{dT}{dt} = -P_{\rm b} + P_{\rm J} + P + \text{Noise} \tag{2}$$

$$L\frac{dI}{dt} = V - IR_{\rm L} - IR(T,I) + \text{Noise} \tag{3}$$

In the model, we use a linear resistance model,

$$R(T,I) = R_0 + \left.\frac{\partial R}{\partial T}\right|_{I_0}(T-T_0) + \left.\frac{\partial R}{\partial I}\right|_{T_0}(I-I_0) \tag{4}$$

where the partial derivatives are described with the parameters

$$\alpha = \left.\frac{\partial \log R}{\partial \log T}\right|_{I_0} = \frac{T_0}{R_0}\left.\frac{\partial R}{\partial T}\right|_{I_0} \tag{5}$$

$$\beta = \left.\frac{\partial \log R}{\partial \log I}\right|_{T_0} = \frac{I_0}{R_0}\left.\frac{\partial R}{\partial I}\right|_{T_0} \tag{6}$$

The model also includes the thermal link to the thermal bath. The noise sources included in the model are thermal fluctuations between the TES and the heat bath, electrical Johnson noise of the TES and shunt resistor, and readout noise from the SQUID and amplifier chain. Also included is an unexplained noise parameter (based on empirical characterization) that represents sources of noise internal to the TES that are as yet not fully understood.
Input parameters of the model are the heat capacity of the sensor, the properties of the thermal link, the operating point resistance and temperature, and depending on whether the TES is AC- or DC-biased, the filter inductance and parasitic resistance, or the shunt resistance and the effective inductance of the readout circuit. These input parameters can either be given on the command line using the Parameter Interface Library (PIL) syntax, or can be read from a FITS file with a `TESDATASTREAM` extension whose header contains the parameters. Such a file can be obtained by running `tessim` with the `propertiesonly=yes` option. As an example, Fig. 12 displays pulse shapes as calculated with `tessim` for the same pixel at different energies, highlighting the non-linear aspect of these simulations.
`tessim` also incorporates an optional trigger stage to extract useful pulse records from the complete simulated data stream. Four options are currently available:

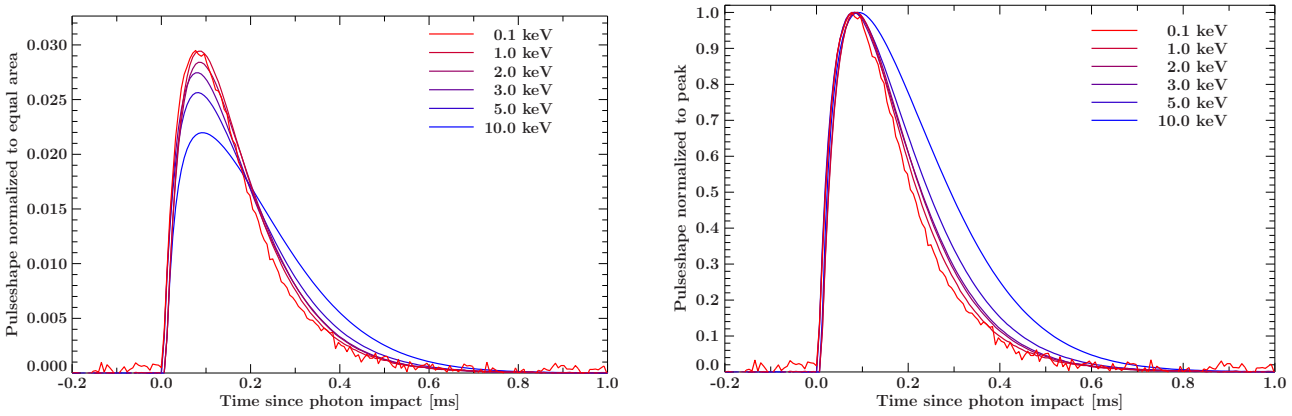

Figure 12: Calculated pulse shapes for an example TES pixel for photons of energy 0.1, 1, 2, 3, 5, and 10 keV, illustrating the variability of the pulse profiles. *Left:* Pulses normalized to the same area. *Right:* Pulses normalized to the same peak current.

- `stream` No trigger is applied and the whole data stream is dumped.
- `noise` Similar to the stream option, the whole data stream is dumped. The difference to `stream` is that the data are dumped as individual records, which is more suitable for the determination of the baseline noise than chopping a data stream by yourself.
- `movavg:npts:threshold:suppress` Use a simple moving average algorithm to trigger once the moving average of the digitized signal is above `threshold`. The number of points used for the average is defined by `npts` and `suppress` defines the minimum number of samples to wait from one trigger until the next trigger is allowed to happen. This latter parameter can be used to ensure that multiple triggers do not happen during one pulse.
- `diff:npts:threshold:suppress` Parametrized similarly as for the previous option but uses the low-pass filtered derivative of the stream instead of the average. `npts` is the number of points that are included in the calculation of the derivative and has to be $2 \leq$ `npts` ≤ 10.

8.3.2 SIRENA

`SIRENA` is the software designed to evaluate the performance of different reconstruction algorithms in order to select the most appropriate for the *Athena* X-IFU on board event processor. It has been fully integrated into the SIXTE software and can extract, from simulated pulse records, the arrival time and energy of the events with a resolution representative of the future instrument performance. This tool then writes an output Event File in the same format as `sixtesim`.
The standard algorithm used by`SIRENA` is the optimal filtering technique (Szymkowiak et al., 1993) but other algorithms were also implemented and are under study. Full information can be found at the official `SIRENA` documentation page: `https://sirena.readthedocs.io`.
In addition, the SIXTE distribution (under `scripts/SIRENA` folder) also incorporates some shell scripts to guide the user to perform the reconstruction of `tessim`-simulated files with `SIRENA`. The three basic steps to be performed are:

1. *Noise spectrum creation*: For different reconstruction methods and in particular, for the optimal filtering technique, a noise spectrum is required. For this purpose, the first step is the simulation of a noise stream (free of pulses) to later build its spectrum:

1.1 Production of noise stream impactlist (`tesconstpileup` or `tesgenimpacts`)

1.2 Simulation of noise stream (`tessim`)

1.3 Creation of the noise spectrum (`gennoisespec`)

2. *Library of filter(s) creation*: At this step a library of optimal filter(s) and (optionally) other pre-calculated values must be created to perform, afterwards, the reconstruction of the simulated data. The optimal filter is calculated using the noise spectrum from step 1 above and an average template of many pulses of a specific energy (several filters at different energies can also be calculated).

   2.1 Production of isolated pulses impact list (`tesconstpileup` or `tesgenimpacts`)

   2.2 Simulation of isolated pulses (`tessim`)

   2.3 Creation of library of optimal filters (`tesreconstruction` with `opmode=0`)

3. *Data reconstruction* (`tesreconstruction` with `opmode=1`): User simulated data can then be reconstructed selecting the reconstruction method as well as several other parameters that can be specified in the `tesreconstruction` call, as for example, the filter length.

9 Simulation calibration

Detectors possess a lower energy threshold, which determines the minimal energy a single pixel must receive to trigger a detection. This threshold minimizes the noise at low energies. Any information of events below that threshold is disregarded and lost for future processing. As a consequence the reconstruction of the photon energy includes a systematic underestimation. In addition to this shift in energy the photon distribution is slightly deformed with respect to that in the corresponding detector response. This systematic error influences the spectral shape and will cause the simulated spectrum to differ from the expected shape (see top ratio panel in Fig. 13).
As this systematic error depends on the size and distribution of the charge cloud and therefore depends on the event pattern and event energy, an individual correction for each detector (and detector setting) is required. SIXTE provides a correction for this systematic error, which is divided into two steps.
In the first step PI values are calculated, which are corrected for the energy shift. This correction is achieved by providing a Pha2Pi correction file, which is a lookup table containing corrected energies for each PHA channel (of the corresponding detector response) and for each event pattern type. In the second step the deformation of the photon distribution is accounted for by a simulated Monte Carlo PI-RMF, which corrects the residual deviations not addressed by the energy shift correction.
An additional correction is necessary in cases, in which the PSF exceeds the spatial extent of the detector. As the PSF is energy dependent the correction is not a simple constant factor. This energy dependent loss of photons can be easily accounted for by calibrating the ARF.
In the following a brief overview of the calibration process of simulated events is given. Section 9.1 describes the automated Pha2Pi correction process, while Sect. 9.2 explains how this correction can be performed manually. In Sect. 9.3 an analysis example is given, which utilizes the Monte Carlo to obtain calibrated simulated spectra.

9.1 Automated Pha2Pi correction

In case the XML used for simulation contains the field

```
<pha2pi filename="pha2pi.fits"/>
```

specifying the Pha2Pi file, the energy shift correction will be performed automatically. The automated correction is run by SIXTE[17] `sixtesim`, but not e.g. `gendetsim`. This tool will produce event files, which contain a `PI` column providing the energy shift corrected PI values in addition to the uncorrected PHA values.
If a PI-RMF file is specified in the XML by

```
<pirmf filename="pirmf.rmf"/>
```

this RMF will be used by analysis tools like `makespec` instead of the standard RMF used for the simulation, which is referenced by the header key `RESPFILE`.
In some cases a calibrated ARF is specified in the XML with

```
<specarf filename="spectral_analysis.arf"/>
```

which accounts for possible energy dependent photon loss rates, e.g., in a defocused detector configuration.
Corresponding to the specified corrections, the header key words `PHA2PI`, `PIRMF`, and `SPECARF` will be added to the event file. The key `PHA2PI` references the pha2pi file used for the energy shift correction and `PIRMF` states the corresponding PI-RMF to be used in further analysis of the PI values (see Sect. 9.3). The header key word `SPECARF` is added to specify the calibrated ARF, if needed.

[17] After version 2.5.1

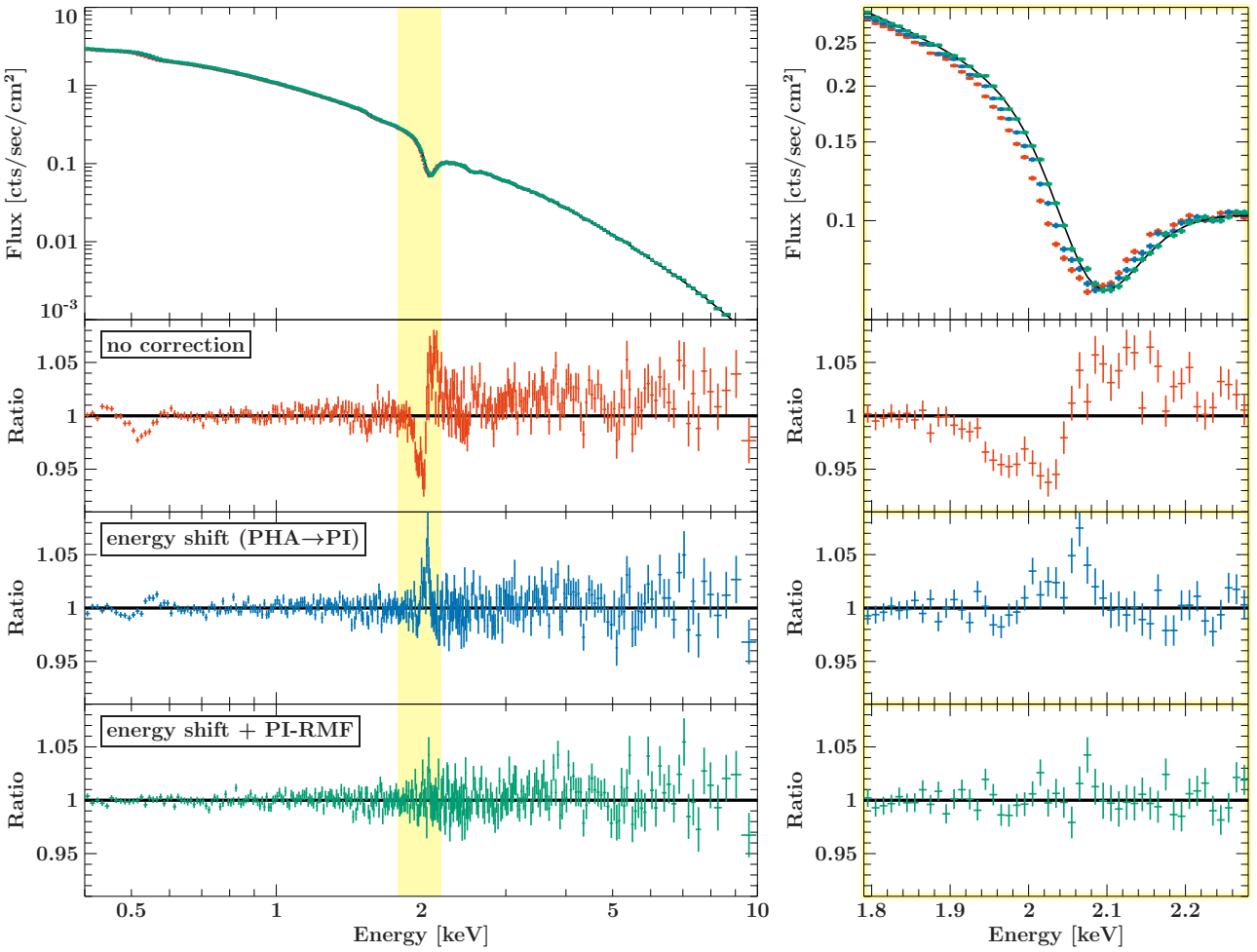

Figure 13: Simulated *Athena*-WFI powerlaw spectra in three different configurations. **Red**: Uncorrected spectrum based on PHA values and nominal RMF used for simulation. **Blue**: Energy shift corrected spectrum based on PI values and nominal RMF. **Green**: PI spectrum utilizing the PI-RMF. Top panels show the spectra, while the bottom panels show the ratio of the data to the input powerlaw spectrum.

9.2 Manual Pha2Pi correction using the `pha2pi` tool

The SIXTE tool `pha2pi` allows to manually apply the correction to a given event file. To perform the correction the corresponding Pha2Pi correction file has to be specified as well as the response file, which was used in the simulation of the event file. The header key words `PHA2PI` and `PIRMF` will be added to the event file, too.

Tool parameters:
EvtFile: Event file, which is to be corrected
Pha2Pi: Pha2Pi correction file, which is to be used to obtain the PI corrected values
RESPPath: Path to the response file
RESPfile: Response file (the same used in the simulation is set by default)

9.3 Monte Carlo PI-RMF

The energy shift corrected PI values of events include only the first part of the calibration. When analysing these PI values, e.g., creating spectra, the second calibration part has to be considered, which is provided by the Monte Carlo RMF.
Figure 13 illustrates the different correction steps. The data shown represent a point source with a simple powerlaw with a photon index of 2, without background and filtered for pile-up simulated for *Athena*-WFI. Based on these data, three different spectra are produced. The first spectrum is based on the uncorrected PHA values in combination with the nominal RMF used for the simulation (red data; top ratio panel). The data show significant deviations from the expected powerlaw input model, especially at the Silicon edge (~ 2 keV). Here the main effect is the energy shift. The middle ratio panel shows the PI spectrum in combination with the nominal RMF, which already shows significant less deviations. The residuals at 2 keV, however, cannot be explained with an energy shift. These residuals result from deformation of the photon distribution, which have to be corrected using the corresponding Monte Carlo RMF as shown in the bottom ratio panel.
When creating spectra with the SIXTE tool `makespec`, it will automatically use PI values if present in the given event file and the corresponding Monte Carlo PI-RMF, which is stored in the header key `PIRMF`.

9.4 ARF calibration

In some cases a calibrated ARF is necessary when analysing the simulated data to reproduce the input as, for example, when simulating a defocused configuration. If the defocused PSF exceeds the spatial extent of the detector, a certain fraction of the photons is not detected. The extent of the defocused PSF can strongly depent on the energy as shown in Fig. 14 and therefore also fraction of lost photons. To account for this effect the ARF used for the simulation has to be corrected with this photon loss fraction to provide a calibrated ARF for further analysis.
The correction factor for each energy is determined by the integrated fraction of the corresponding PSF, which is outside of the detector area. These factors are linearly interpolated onto the ARF energy grid. An example for this ARF calibration is shown in Fig. 14.
Aside from this example, there are also other effects that can affect the ARF of a particular observation:

- Vignetting, when a source is not exactly on-axis
- Pixel gaps, such as in the *Athena X-IFU*
- Chip gaps, such as in the *Athena WFI*
- Non-constant pointings, for example due to dithering or slewing, which can cause all of the above effects in a time-variable way.

As such, outside of very simple observations, an ARF should be generated using the tool `sixte_arfgen` when fitting extracted spectra.

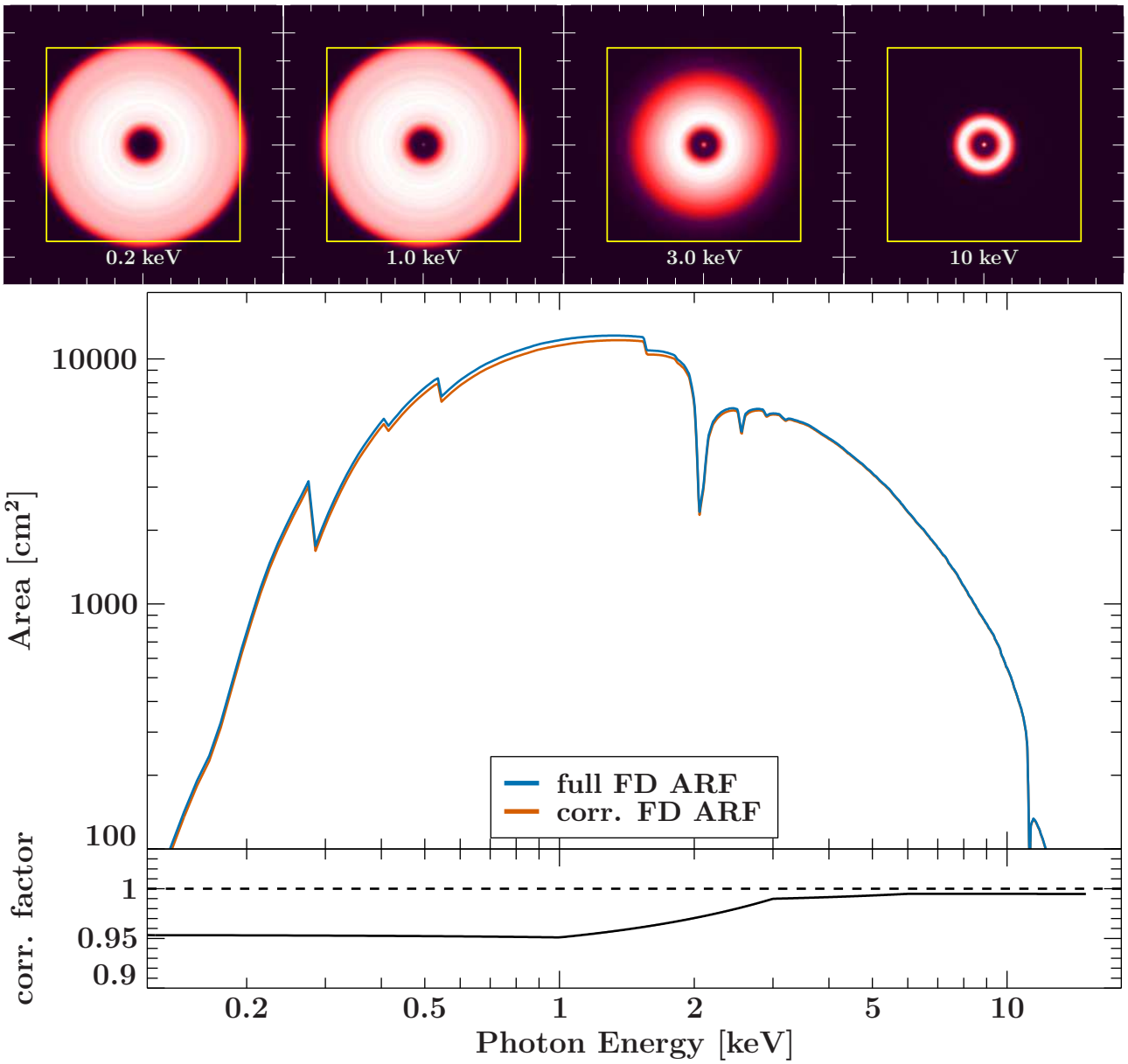

Figure 14: **Top:** Defocused PSF of the *Athena*-WFI at different energies. The spatial extent of the *Athena*-WFI fast detector is outlined in yellow. **Bottom:** Full ARF of the *Athena*-WFI fast detector (blue) and the corresponding corrected ARF accounting for the energy dependent photon loss.

10 Tutorial and Cookbook for SIXTE Simulations

10.1 Introduction

In this section we present typical use-cases of SIXTE using different science simulations for *Athena* as examples. We start in Sect. 10.2 with a general overview of simulations of simple point sources, where we also introduce other tools from the HEASOFT package that allow an in-depth inspection of the data files produced by SIXTE. We then continue in Sect. 10.4 with example simulations of deep cosmological fields for the *Athena*-WFI, continue in Sect. 10.5 with examples for simulating extended sources in, and end in Sect. 10.7 with simulations of high resolution spectra for the *Athena*-X-IFU.
Please note that each of these sections is self-contained, such that the explanation of some initial commands is repeated multiple times.

10.2 General Introduction to Simulations with SIXTE

As discussed in more detail in Sect. 1, SIXTE is a Monte Carlo based tool that can simulate observations of astrophysical sources for a wide variety of different current and future X-ray astronomical satellites. A SIXTE simulation consists of three steps:

1. *Preparation of the input of the simulation, using so-called SIMPUT files (Sect. 2):* In this step we define what is to be observed. Typically this will be a pointed observation of a field on the sky. In the most simple case, this field will only contain one point source, but the simulations also allow to take into account extended sources, time variability, or the simulation of large catalogues of astronomical sources that can contain millions of X-ray sources on the whole sky. Later examples in this and the following sections will present some of the more advanced features.
2. *Running the simulation:* In this step photons from all sources that are visible to the instrument are generated using a Monte Carlo algorithm (Sect. 3). These photons are then projected onto the focal plane of the instrument using a model of the instrumental optics (Sect. 4) where they are detected using one of the available instrumental models (Sect. 6).
3. *Analyzing the simulation:* The output of the previous step are one or multiple standard FITS event files whose structure follows X-ray astronomical standards as specified by NASA's HEASARC. The output can therefore be analyzed using standard astronomical data analysis packages. SIXTE provides some tools that prepare standard data products such as spectra and images from event files, alternatively other tools that can read FITS event files can be used.

Before we start in the next sections with more complicated examples, in the following we will illustrate how a SIXTE simulation works using a simple observation of a point source.

10.2.1 Step 0: SIXTE installation setup

Before starting any simulation make sure SIXTE is set up correctly. The complete instructions for the installation and download of all necessary files can be found on the SIXTE homepage in the installations section[18].
After the installation it is important to initialize SIXTE. For the `(t)csh`, the necessary commands would be

```
setenv SIXTE sixtedir
source ${SIXTE}/bin/sixte-install.csh
```

where `sixtedir` is the installation directory for SIXTE and the SIMPUT tools. These are best put in the `~/.cshrc` file. If you use `bash`, the commands are

[18] `https://www.sternwarte.uni-erlangen.de/sixte/installation/`

```
export SIXTE=sixtedir
. ${SIXTE}/bin/sixte-install.sh
```

Again, it is best to include these commands in the shell's initialization file (typically `~/.bashrc`). We assume that HEASOFT is already initialized at this point.

10.2.2 Step 1: Preparing the SIMPUT-file

The first and most important step in doing a simulation for an instrument is the description of what we want to observe. The more realistic this description, the more realistic the output of our simulation. In order to allow us to perform the best simulations possible, SIXTE bases the simulations on a FITS standard called SIMPUT. SIMPUT-files allow us to specify point sources, but they also scale well to very complicated setups. We will give several examples for such setups in later sections of this tutorial.
While SIMPUT files can be generated from scratch using your own scripts, for many simulations it is easier to build the SIMPUT file using the program `simputfile`. Like all SIXTE programs, `simputfile` does not provide a GUI, but it runs as a command line tool that uses the so-called `parfile`-interface, similar to the well-known HEASOFT software.
Let us first take a look at the parameters provided by `simputfile` using the FTOOL `plist`:[19]

```
computer:~/> plist simputfile
Parameters for /home/wilms/pfiles/simputfile.par
       Simput = simput.fits      output SIMPUT catalog file
      (Src_ID = 1)               source ID
    (Src_Name = none)            source name
           (RA = 0.0)            right ascension (deg)
          (Dec = 0.0)            declination (deg)
     (srcFlux = 0.0)             source flux (erg/s/cm^2)
         (Elow = 0.1)            lower bound of the generated spectrum (keV)
          (Eup = 100.0)          upper bound of the generated spectrum (keV)
        (Estep = 0.0)            deprecated (resolution of the spectrum (keV) )
        (Nbins = 1000)           number of energy bins created from Elow to Eup
    (logEgrid = no)              use a logarithmic energy grid (from Elow to Eup with Nbins)
  (plPhoIndex = 2.0)             power law index
       (plFlux = 0.0)            power law flux (erg/s/cm^2)
         (bbkT = 1.0)            black body temperature (keV)
       (bbFlux = 0.0)            black body flux (erg/s/cm^2)
     (flSigma = 1.0e-3)          Fe line sigma (keV)
       (flFlux = 0.0)            Fe line flux (erg/s/cm^2)
     (rflSpin = 0.0)             rel. Fe line spin
     (rflFlux = 0.0)             rel. Fe line flux (erg/s/cm^2)
           (NH = 0.0)            N_H (10^22 atoms/cm^2)
         (Emin = 1.0)            E_min of reference energy band for fluxes (keV)
         (Emax = 10.0)           E_max of reference energy band for fluxes (keV)
    (ISISFile = none)            ISIS spectral parameter file (*.par)
    (ISISPrep = none)            Additional ISIS script to be executed before loading parameter file.
   (XSPECFile = none)            XSpec spectral model file (*.xcm)
   (XSPECPrep = none)            Additional XSPEC script to be executed before loading parameter file.
     (PHAFile = none)            PHA spectrum
   (ASCIIFile = none)            ASCII spectrum
      (LCFile = none)            ASCII file containing a light curve
      (MJDREF = 55000.0)         reference Modified Julian Date
      (PSDnpt = 10000)           number of frequency bins in PSD
```

[19] Programs distributed with HEASOFT are called "ftools". If you want to know more about the parameters of an FTOOL, use the command `fhelp ftoolname` where "`ftoolname`" is the name of the FTOOL. To get a list of all available FTOOLS, use `fhelp ftools`.

```
    (PSDfmin = 1e-8)          minimum frequency in PSD
    (PSDfmax = 1e4)           maximum frequency in PSD
        (LFQ = 0.0)           Quality of low frequency Lorentzian
      (LFrms = 0.0)           rms of low frequency Lorentzian
       (HBOf = 0.0)           peak frequency of horizontal branch Lorentzian (Hz)
       (HBOQ = 0.0)           Quality of horizontal branch Lorentzian
     (HBOrms = 0.0)           rms of horizontal branch Lorentzian
       (Q1f = 0.0)            peak frequency of 1st QPO Lorentzian (Hz)
       (Q1Q = 0.0)            Quality of 1st QPO Lorentzian
     (Q1rms = 0.0)            rms of 1st QPO Lorentzian
       (Q2f = 0.0)            peak frequency of 2nd QPO Lorentzian (Hz)
       (Q2Q = 0.0)            Quality of 2nd QPO Lorentzian
     (Q2rms = 0.0)            rms of 2nd QPO Lorentzian
       (Q3f = 0.0)            peak frequency of 3rd QPO Lorentzian (Hz)
       (Q3Q = 0.0)            Quality of 3rd QPO Lorentzian
     (Q3rms = 0.0)            rms of 3rd QPO Lorentzian
   (PSDFile = none)           ASCII file containing a PSD
(ImageFile = none)            FITS file containing an image of the spatial flux distribution
   (chatter = 3)              verbosity
   (clobber = no)             overwrite output files if exist?
   (history = true)           write a history block with program parameters to each FITS file?
```

The number of parameters is very large as `simputfile` can be used to setup rather complex SIMPUT-files. A full description of the parameters of all SIMPUT tools is given in Sect. A. In the output of `plist` arguments that are *not* given in parentheses are mandatory, arguments that are listed in parentheses are not mandatory. The output contains the name of the argument, its default value after the =-sign, and a brief description of the meaning of the parameter. The elegance of the parfile interface is that the sequence with which you give the arguments does not matter, since all arguments are identified by their respective parameter name.[20]

We will now generate a SIMPUT-file for a non-variable source with a spectrum that can be described by a simple absorbed power law. Inspection of the list above shows that we can in principle specify the source spectrum with the `plPhoIndex`, `plFlux`, and `NH` parameters. We will not do this here (this is left as an exercise to the reader), but rather we will use a more powerful interface: `simputfile` can read model parameter files produced by the X-ray data analysis packages `XSPEC`[21] and `isis`[22].[23] This means that as long as you have a best fit model for a source available (e.g., from the analysis of archival data), you can use it to generate a SIMPUT-file.

We will now use `XSPEC` to prepare an `xcm`-file for a source that has an absorbed power law spectrum with an unabsorbed flux of 2.16×10^{-11} erg cm^{-2} s^{-1}, a photon index of $\Gamma = 2.05$, and a foreground absorption with an equivalent hydrogen column of 2×10^{21} cm^{-2}:[24]

```
computer:~/> xspec

                XSPEC version: 12.9.0i
        Build Date/Time: Tue Mar 22 23:19:59 2016

XSPEC12>model phabs*pegpwrlw

Input parameter value, delta, min, bot, top, and max values for ...
```

[20] This is not entirely correct – mandatory arguments can be given without their argument name. We strongly recommend not doing this.

[21] `https://heasarc.gsfc.nasa.gov/xanadu/xspec/`

[22] `http://space.mit.edu/asc/isis/`

[23] XSPEC-files are called `xcm`-files, as XSPEC appends `xcm` to the files. ISIS appends `par`. Do not confuse these latter files with parfiles produced by HEASOFT or SIXTE.

[24] This happens to be a source with roughly the same spectral shape as the Crab nebula and a flux of 1 mCrab.

```
              1      0.001(      0.01)          0          0     100000       1e+06
1:phabs:nH>0.2
              1       0.01(      0.01)         -3         -2          9          10
2:pegpwrlw:PhoIndex>2.05
              2      -0.01(      0.02)       -100       -100       1e+10       1e+10
3:pegpwrlw:eMin>2
             10      -0.01(       0.1)       -100       -100       1e+10       1e+10
4:pegpwrlw:eMax>10
              1       0.01(      0.01)          0          0       1e+20       1e+24
5:pegpwrlw:norm>21.6

========================================================================
Model phabs<1>*pegpwrlw<2> Source No.: 1   Active/Off
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    0.200000     +/-  0.0
   2    2   pegpwrlw   PhoIndex            2.05000      +/-  0.0
   3    2   pegpwrlw   eMin       keV      2.00000      frozen
   4    2   pegpwrlw   eMax       keV      10.0000      frozen
   5    2   pegpwrlw   norm                21.6000      +/-  0.0
------------------------------------------------------------------------

XSPEC12>flux 2 10
 Model Flux 0.0033239 photons (2.1374e-11 ergs/cm^2/s) range (2.0000 - 10.000 keV)
XSPEC12>save model mcrab.xcm
XSPEC12>quit
```

The xcm-file looks as follows:

```
method leven 10 0.01
abund wilm
xsect bcmc
cosmo 70 0 0.73
xset delta 0.01
systematic 0
model  phabs*pegpwrlw
            0.2      0.001          0          0     100000       1e+06
           2.05       0.01         -3         -2          9          10
              2      -0.01       -100       -100       1e+10       1e+10
             10      -0.01       -100       -100       1e+10       1e+10
           21.6       0.01          0          0       1e+20       1e+24
```

It is easiest to use XSPEC to see what the spectrum looks like:

```
XSPEC12>cpd /xs
XSPEC12>dummyrsp 0.1 15 1000 log
XSPEC12>plot model
```

This sequence of commands opens the `/xs`-plotting window, defines a diagonal response matrix from 0.1 to 15 keV with 1000 energy bins on a logarithmic grid, and plots the model. If you have already exited XSPEC since you typed in the `quit` command, reenter XSPEC first and use the command

```
XSPEC12> @mcrab.xcm
```

to load the model description again.

Exercise: Add a narrow Gaussian iron line at an energy of 6.4 keV with an equivalent width of 150 eV to the spectrum.
Hint: use the `editmod` command to add a Gaussian line to the model. "Narrow line" means 1 eV or so. To get the equivalent width right, first chose a small value for the normalization. Then use the `eqw` command of XSPEC to determine the equivalent width of this initial guess and change the norm appropriately using XSPEC's `newpar` command.

We are now ready to generate the SIMPUT file. As mentioned above, the SIMPUT format was built such that it scales well to large number of sources. It would be prohibitive for disk space if we had to specify the spectral shape of all of these sources. SIMPUT therefore allows us to use a single spectral shape for multiple sources. For this reason `simputfile` allows us to override the flux that is encoded in the `xcm`-file when generating the SIMPUT file. Due to the larger number of parameters of `simputfile`, we recommend that you write a small shell script to generate the SIMPUT file. This will also allow you to copy and modify the script for the exercises later on. The shell script should look as follows:

```
#!/bin/bash

base=mcrab
$SIXTE/bin/simputfile Simput=${base}.fits \
                      Src_Name=first \
                      RA=0.0 \
                      Dec=0.0 \
                      srcFlux=2.137e-11 \
                      Elow=0.1 \
                      Eup=15 \
                      NBins=1000 \
                      logEgrid=yes \
                      Emin=2 \
                      Emax=10 \
                      XSPECFile=${base}.xcm
```

Note the "\ " at the end of each line. These tell the Unix-shell that the line is continued on the next line. To save us some typing later we place our source at a position of $\alpha = 0.000°$ and $\delta = 0.000°$.
Edit the script with your favorite editor, save it as `mcrab.bash`, make it executable (`chmod ugo+x ./mcrab.bash`), and execute it.[25] You can see from the output that `simputfile` executes XSPEC in order to get the photon flux as a function of energy on the energy grid specified by the `Elow` and `Eup` arguments.

[25] If you have worked with the extraction of *Chandra* data before, you might know about the `pset` command, which allows to change the default values of hidden parameters. The *Chandra* X-ray center recommends in its threads to set such hidden parameters and then to call FTOOLS without explicitly listing these parameters. We recommend against this approach. While it results in shorter command lines, it also can lead to users forgetting that they changed something, leading to unexpected results. The approach is also prone to race conditions when running multiple simulations or extractions in parallel. In case something goes wrong with any FTOOL or SIXTE command, the first thing is always to reset its parameters. For this you can either use the `punlearn` command or just remove the contents of the local copy of the parfiles, the default location of which is in `~/pfiles` (or rather: in the first directory listed in the environment variable `PFILES`). In other words, do a "`rm ~/pfiles/*`".

If all went ok, you should now have a SIMPUT-file named `mcrab.fits` in your directory:

```
computer:~/> ls -l
total 36
lrwxrwxrwx 1 sixte sixte   280 Apr 17 20:17 mcrab.bash
-rw-r--r-- 1 sixte sixte 28800 Apr 17 20:18 mcrab.fits
-rw-r--r-- 1 sixte sixte   465 Apr 17 19:59 mcrab.xcm
```

10.2.3 An Aside: Inspecting FITS-files with HEASOFT tools

Before we continue with the description of the simulation, we will first introduce a few HEASOFT commands that will come in handy later, when inspecting the output of the simulations. If you are already familiar with HEASOFT and FITS-files, you can skip this section.
In the previous section we generated a new SIMPUT file, `mcrab.fits`. We will now inspect its contents. First, we take an overview at the general structure of the file:

```
computer:~/> fstruct mcrab.fits
  No. Type     EXTNAME      BITPIX Dimensions(columns)      PCOUNT  GCOUNT

   0  PRIMARY                 16     0                          0    1
   1  BINTABLE SRC_CAT         8     168(12) 1                  0    1

      Column Name               Format     Dims       Units     TLMIN  TLMAX
      1 SRC_ID                    J
      2 SRC_NAME                  32A
      3 RA                        D                  deg
      4 DEC                       D                  deg
      5 IMGROTA                   E                  deg
      6 IMGSCAL                   E
      7 E_MIN                     E                  keV
      8 E_MAX                     E                  keV
      9 FLUX                      E                  erg/s/cm**2
     10 SPECTRUM                  32A
     11 IMAGE                     32A
     12 TIMING                    32A

   2  BINTABLE SPECTRUM        8     64(3) 1                 11920    1

      Column Name               Format     Dims       Units     TLMIN  TLMAX
      1 ENERGY                    1PE(1490           keV
      2 FLUXDENSITY               1PE(1490           photon/s/cm**2/keV
      3 NAME                      48A
```

FITS files consist of individual tables, which are called FITS-extensions. Each of these extensions consists of a header in which various parameters are defined and the data table. In addition, a FITS file has its own header, called the `PRIMARY` header.
Our SIMPUT-file therefore contains two extensions, which are named `SRC_CAT` and `SPECTRUM` and are of type `BINTABLE` (this means they are binary tables)
The extension `SRC_CAT` contains the source catalogue. The output of fstruct tells us that each row of the catalogue has a length of 168 bytes and it contains 12 columns ("`168(12)`"). It has only one row (the 1 following `168(12)`). The other information in this line is not important for us. The information about this extension is followed by the information about the columns in the table. As we just discussed, this catalogue has 12 columns, `SRC_ID`, `SRC_NAME`,...,`TIMING`. The names of these columns are pretty much self explanatory. Their data type

is given by the column `Format` in the listing above. FITS uses Fortran data types, which are explained in the following list:
`J`: a 16 bit integer
`32A`: an ASCII-string of 32 characters length
`D`: a double precision floating point number
`E`: a single precision floating point number
Note that only the position of the source (columns `RA` and `DEC`) is saved in double precision, all other columns are in single precision to save space.
Now that we have seen the structure of the file, we can take a look at its contents. There are several programs to do so. An interactive tool is the FITS-viewer `fv`, but often it is faster and easier to use the command line tool `fdump`. Both programs are distributed with HEASOFT. Of special interest are the `SPECTRUM`, `IMAGE`, and `TIMING` columns, which contain information about the source spectrum, the image, and the variability of the source. These columns contain strings which can point at other files or at other extensions within the file. Inspecting `mcrab.fits` with `fdump` shows you that the `IMAGE` and `TIMING` have the value `NULL`, that is, the source is a point source (no image is present) and is not variable (no timing-information is present). The value of the `SPECTRUM` column is `[SPECTRUM,1]`, this means that the X-ray spectrum of this source is defined by the first `SPECTRUM`-extension in this file. Alternatively, this column could also include a path-name to another file. Looking at the structure of the `SPECTRUM`-extension shows you that the extension has the columns `ENERGY`, `FLUXDENSITY`, and `NAME`. `ENERGY` and `FLUXDENSITY` contain the flux density at a given energy, and are used by SIXTE to interpolate the X-ray spectrum to the resolution necessary for the simulations. `NAME` can be used to identify the spectrum using the extended FITS name syntax (see below)

Exercise: Generate a second SIMPUT file for our source, using the `plPhoIndex`, `plFlux`, and `NH` parameters rather than an `xcm`-file. Compare both SIMPUT-files using `fv`. Note this step only works if you have `ISIS` installed.

10.2.4 Step 2: Running the simulation

Once a simputfile exists, we can run the simulation. All properties describing an instrument are set in XML-files. These are human readable files that describe, e.g., the pixel size, the quantum efficiency, and other properties of the detector (see Sect. 6).
The general simulator is called with `sixtesim`, and can perform a simulation for any given XML-file, including the full four chip *Athena*-WFI (see tutorial in Sect. 10.4) and the *Athena*-X-IFU (see tutorial in Sect. 10.7). The instrument files are available at `https://www.sternwarte.uni-erlangen.de/sixte/example-source-files/`.
To run a simple simulation of the WFI, we use the `sixtesim` tool and the XML-file for one large chip of the WFI. We have to specify the location of the XML-files[26] that make up the detector and then point the detector at the proper location on the sky. This is done with the following `bash`-script:

```
#!/bin/bash

base=mcrab

xmldir=$SIXTE/share/sixte/instruments/athena-wfi/wfi_wo_filter
xml=${xmldir}/ld_wfi_ff_large.xml

$SIXTE/bin/sixtesim \
```

[26] Have a look at Tab. 1 in Sect. 6.1 for a list of available XML-files for the WFI.

```
    XMLFile=${xml} \
    RA=0.000 Dec=0.000 \
    Prefix=sim_ \
    Simput=${base}.fits \
    EvtFile=evt_${base}.fits \
    Exposure=1000
```

Make the script executable and then execute it. You will get a warning that the spectrum in the SIMPUT-file does not cover the full ARF. This is typically the case when the parameters `Elow` and `Eup` of `simputfile` were chosen too conservatively.
In the call to `sixtesim`, note especially the values of the `RA` and `Dec` arguments. These describe the pointing direction. Since our source is at $\alpha = 0.000°$ and $\delta = 0.000°$, we are performing an on-axis observation. Note that if you are simulating real sources, it is good practice to place the source in the SIMPUT file at its proper location. This will simplify your life later when you want to merge different SIMPUT-files, e.g., to produce a SIMPUT-file for a more complicated source region using SIMPUT-files for individual regions. Furthermore, you can speed up the simulation by altering the `dt` qualifier, which gives the time increment of the attitude file in seconds. Note that this is not the time resolution of the detector (which is instead given in the XML file).
Finally, after the initial simulation is done (the counter increases to 100%), you see messages that the event pattern analysis is done. During this step charges detected in individual pixels on the chip are combined into a single photon event. These events are then projected back onto the sky.

Exercise: Modify your SIMPUT-file to cover the full energy range of the ARF.
Note that you actually have to go slightly past the limits listed in the ARF because the energy grid of the spectrum is generated at the midpoints of the bins from `Elow` to `Eup`.

10.2.5 Step 3: Analyzing the simulation

We first look at the event files produced during the simulation. Just looking at their length we notice that only chip0 contains any data. This is expected: the WFI is designed such that the optical axis is located on chip0. We therefore only analyze this event file further.

Exercise: Use `fstruct` and `fv` or `fdump` to take a look at the structure of the event file. How many events were simulated? Speculate on the meaning of the individual columns in the event file (see below for an explanation of their definition).

We first generate an image using `imgev`:

```
$SIXTE/bin/imgev \
    EvtFile=sim_evt_mcrab.fits \
    Image=img_mcrab.fits \
    CoordinateSystem=0 Projection=TAN \
    NAXIS1=512 NAXIS2=512 CUNIT1=deg CUNIT2=deg \
    CRVAL1=0.0 CRVAL2=0.0 CRPIX1=256.5 CRPIX2=256.5 \
    CDELT1=-6.207043e-04 CDELT2=6.207043e-04 \
    history=true clobber=yes
```

The detailed arguments of the command are explained in the next section. In brief, the `Projection`-argument defines the map projection that is to be used. All projections allowed by the FITS standard are allowed here. For most applications and for pointed observations, the tangential projection, `TAN`, is recommended. The

NAXIS-keywords give the number of x- and y-pixels in the picture. CRVAL1 and CRVAL2 define the center of the map. They should in general equal the pointing direction of the simulated observation. CRPIX1 and CRPIX2 are the point corresponding to the optical axis on the instrument. These depend on the instrument that is being simulated. Finally, CDELT1 and CDELT2 define the spatial resolution of the detector (that is, one detector pixel on the WFI has a size of about $2.2'' \times 2.2''$).
The image produced by `imgev` can be viewed with standard astronomical image viewers such as `ds9`[27] or `fv`. In our current case, it is rather boring – we just have a point source.
More interesting for point sources is how well the spectral shape of the source can be recovered from the observation. In order to produce the X-ray spectrum we can use the tool `makespec`. Its arguments are the event file, the name of the spectral file, a selection expression describing what events should be binned into the spectrum, and a path to the response matrix.

```
$SIXTE/bin/makespec \
    EvtFile=sim_evt_mcrab.fits \
    Spectrum=spec_mcrab.pha \
    EventFilter="(RA>359.95 || RA<0.05) && Dec>-0.05 && Dec<+0.05" \
    RSPPath=${xmldir} clobber=yes
```

where `${xmldir}` was defined in the script above. Note that setting the option `clobber=yes` simply overwrites existing files by default.
The `EventFilter` can perform selections on all columns of the FITS file. The selection expression uses the standard C-syntax for logicals, i.e., `&&` means a logical AND, `||` means a logical OR, and parentheses can be used. SIXTE uses the standard selection syntax offered by the cfitsio-library. Use the command `fhelp rowfilter` to obtain more information about this syntax.
A `ds9` region file can also be used to perform spatial filtering. For this, you first have to add X,Y sky coordinates and a WCS to the event file with the `radec2xy` tool

```
$SIXTE/bin/radec2xy \
    EvtFile=sim_evt_mcrab.fits \
    projection=TAN \
    RefRA=0 RefDec=0
```

where `RefRA` and `RefDec` define the WCS reference point (which should in general correspond to the pointing direction), and `projection` sets the WCS projection type. You can then specify the region file in the `makespec` command with

```
  regfile=ds9.reg
```

You can also select on Vector columns such as `SRC_ID` or `PH_ID`. In this case you need to give a valid logical expression, by selecting only one element of the array.[28] *Important*: the array index starts at '1'. In order to select only background events, the expression would read[29]

```
  EventFilter="SRC_ID[1]==-1"
```

[27] `http://ds9.si.edu`

[28] More information can be found in the `cfitsio` manual at `https://heasarc.gsfc.nasa.gov/fitsio/c/c_user/node81.html`.

[29] Note we only check the first and not the second index of the `SRC_ID` field, which could also contain an additional photon, which is especially important for bright sources. Do not combine this filter with a source region filter to extract the background at the source for spectral analysis, since you would completely remove the background from the source spectrum and therefore remove the Poisson fluctuation of the background.

You have now obtained a PHA-file that can be analyzed with standard X-ray astronomical packages[30]. In order to do so we need to know the names of the the response matrix and the ancilliary response matrix (the "RMF" and the "ARF"). These names can be obtained by looking at the values of the `RESPFILE` and `ANCRFILE` keywords in the header of the PHA-file, either using `fdump` or `fv`, or by printing the names of these keywords using the FTOOL `fkeyprint`.

Exercise: Do this!

Having obtained the names, in order to simplify our life we softlink these files into our working directory:

```
computer:~/> ln -s $xmldir/athena_wfi_pirmf_v20201123.rmf .
computer:~/> ln -s $xmldir/athena_sixte_wfi_wo_filter_v20190122.arf .
```

We can then load the spectrum into XSPEC and analyze it:

```
computer:~/> xspec

                XSPEC version: 12.9.0i
        Build Date/Time: Tue Mar 22 23:19:59 2016

XSPEC12>data spec_mcrab.pha
***Warning: POISSERR keyword is missing or of wrong format, assuming FALSE.

Warning: Statistics not present and Poisson Error Key not set to 'true': setting errors to zer
Warning: RMF TELESCOPE keyword is not consistent with spectrum

1 spectrum  in use

Spectral Data File: spec_mcrab.pha  Spectrum 1
Net count rate (cts/s) for Spectrum:1  1.562e+02 +/- 0.000e+00
 Assigned to Data Group 1 and Plot Group 1
  Noticed Channels:  1-1498
  Telescope: Athena Instrument: WFI  Channel Type: PI
  Exposure Time: 1000 sec
 Using fit statistic: chi
 Using test statistic: chi
 Using Response (RMF) File            athena_wfi_sixte_v20150504.rmf for Source 1
 Using Auxiliary Response (ARF) File  athena_sixte_wfi_wo_filter_v20190122.arf

XSPEC12>cpd /xs
XSPEC12>plot ldata
```

In the above we load the spectrum in (ignoring the warning messages for the moment). We then set the pgplot plotting interface to `/xs` and plot the spectrum in (arbitrary) channels.
To get a better overview of where the features are in the spectrum, we switch the display into energy mode:

[30] As of now SIXTE does not yet include a tool that allows to calculate the `BACKSCAL` parameter or that allows to calculate ARFs for more complicated source regions. This means that for more complicated source regions fluxes cannot yet be recovered. Development of these tools is work in progress.

```
XSPEC12>setplot energy
XSPEC12>plo
```

Note the "ragged" shape at low and high energies, which is due to the small number of photons detected. Note also the feature around 2 keV, which is due to the M-edge of iridium in the mirror material of *Athena*.
To fit the spectrum, we first ignore the energy bands where the signal to noise ratio is small and check that all went ok by plotting again.

```
XSPEC12>ignore **-0.3
    28 channels (1-28) ignored in spectrum #     1

XSPEC12>ignore 4.-**
  1101 channels (398-1498) ignored in spectrum #     1

XSPEC12>plot
```

We then setup an absorbed power law spectrum and fit:

```
XSPEC12>model phabs*power

Input parameter value, delta, min, bot, top, and max values for ...
               1      0.001(      0.01)          0          0     100000      1e+06
1:phabs:nH>
               1       0.01(      0.01)         -3         -2          9         10
2:powerlaw:PhoIndex>
               1       0.01(      0.01)          0          0      1e+20      1e+24
3:powerlaw:norm>

========================================================================
Model phabs<1>*powerlaw<2> Source No.: 1   Active/On
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   phabs      nH         10^22    1.00000      +/-  0.0
   2    2   powerlaw   PhoIndex            1.00000      +/-  0.0
   3    2   powerlaw   norm                1.00000      +/-  0.0
------------------------------------------------------------------------


Fit statistic : Chi-Squared =   4.054716e+11 using 369 PHA bins.

***Warning: Chi-square may not be valid due to bins with zero variance
            in spectrum number(s): 1


Test statistic : Chi-Squared =   4.054716e+11 using 369 PHA bins.
 Reduced chi-squared =   1.107846e+09 for    366 degrees of freedom
 Null hypothesis probability =   0.000000e+00

***Warning: Chi-square may not be valid due to bins with zero variance
```

```
            in spectrum number(s): 1

 Current data and model not fit yet.
```

The warning message regarding the Chi-square means that despite ignoring below 0.3 keV and above 4 keV there are channels in the spectrum which have zero counts. We therefore have to rebin the spectrum using the FTOOL grppha. After exiting XSPEC:

```
computer:~> grppha spec_mcrab.pha spec_mcrab_rebin.pha

  -------------------------
  MANDATORY KEYWORDS/VALUES
  -------------------------
  ------------------------------------------------------------------------
  ------------------------------------------------------------------------
  EXTNAME   - SPECTRUM        Name of this BINTABLE
  TELESCOP  - Athena          Mission/Satellite name
  INSTRUME  - WFI             Instrument/Detector
  FILTER    - Normal          Instrument filter in use
  EXPOSURE  - 1000.0          Integration time (in secs) of PHA data
  AREASCAL  - 1.0000          Area scaling factor
  BACKSCAL  - 1.0000          Background scaling factor
  BACKFILE  - NONE            Associated background file
  CORRSCAL  - 0.0000          Correlation scaling factor
  CORRFILE  - NONE            Associated correlation file
  RESPFILE  - athena_wfi_sixte_v20150504.rmf
  ANCRFILE  - athena_sixte_wfi_wo_filter_v20190122.arf
  POISSERR  - FALSE           Whether Poissonian errors apply
  CHANTYPE  - PI              Whether channels have been corrected
  TLMIN1    - 0               First legal Detector channel
  DETCHANS  - 1498            No. of legal detector channels
  NCHAN     - 1498            No. of detector channels in dataset
  PHAVERSN  - 1.2.1           OGIP FITS version number
  STAT_ERR  - FALSE           Statistical Error
  SYS_ERR   - FALSE           Fractional Systematic Error
  QUALITY   - TRUE            Quality Flag
  GROUPING  - FALSE           Grouping Flag
  ------------------------------------------------------------------------
  ------------------------------------------------------------------------
GRPPHA[] group min 20
GRPPHA[] exit
** UPDPHA 1.0.0 WARNING:       Unknown format: 1.2.1
 ...... Resetting format (HDUVERS) to 1.2.0
 ... written the PHA data Extension
 ...... exiting, changes written to file : spec_mcrab_rebin.pha
 ** grppha 3.0.1 completed successfully
```

This rebins the spectrum to have at least 20 counts in each channel.
Afterwards, the spectrum can be fit in XSPEC:

```
XSPEC12>fit
```

Exercise: Load the rebinned PHA-file into XSPEC and fit an absorbed power law to the 0.3–4 keV band.

Fitting the spectrum will yield a photon index of around $\Gamma = 1.90$, and not $\Gamma = 2.05$ as expected. The reason for this is that the flux of the source is so bright that interesting detector effects become important. The most important here is pile up which is caused by the arrival of more than one photon at the same or neighboring pixels during one read-out cycle of the detector. Two or more photons hitting the same pixel during one frame will produce a valid event if the total charge deposited remains below a predetermined threshold. On the other hand, multiple photons hitting adjacent pixels can produce valid split patterns that are incorrectly identified with a single photon.
In both cases the charge deposited is then converted back to the charge of one event with an energy that is the sum of the energies of the primary events, resulting in a hardening of the spectrum. An advantage of simulations is the possibility to obtain additional information about the emergence of the events which would not be available in real observations. One way of checking whether pileup is important is to take a look at the diagnostic information that is contained in the FITS-file and see whether the `PILEUP`-column has nonzero values. A fast way of doing so is the FTOOL `fstatistic`:

```
computer:~/> fstatistic sim_evt_mcrab.fits PILEUP -
 The sum of the selected column is                  422.00000
 The mean of the selected column is                5.28100715E-03
 The standard deviation of the selected column is   7.24788510E-02
 The minimum of selected column is                  0.0000000
 The maximum of selected column is                  1.0000000
 The number of points used in calculation is           79909
```

The value of `PILEUP` is 1 if more than one photon contributed to the event. In our example, 422 events were piled up (out of 79909). This means that about 0.5% of all events are affected by pile up, so it is no surprise that the spectral shape differs from the input spectral shape.

Exercise: Change the pointing direction of the instrument away from the source in steps of $4'$ in right ascension and declination simultaneously. Take a look at the images of the source and the source count rate. What do you observe?
The decrease in count rate is due to the vignetting of the telescope, while the increase in image size is due to the astigmatism of the Wolter optics (which is not yet modeled correctly in SIXTE due to lack of information about the precise form of the astigmatism of this telescope).

Exercise: In this exercise we continue studying the effects that bright sources have on the measurement process more. Using the Crab-like spectrum defined before, increase the source flux by a factor of 10, 100, and 1000 and redo the WFI simulation. Why can you prepare the simulation using `mcrab.fits` and `fv`, and without running `simputfile`?
Calculate the count rates of the simulations by looking at the number of events in the file. What do you notice? Take a look at the X-ray spectrum of the 2nd brightest simulation and compare the observed photon spectrum with your input photon index.

The simulated event files contain four additional keywords that provide diagnostic information about the origin of the events: The keyword `NVALID` gives the total number of events that were classified as valid in the simulation. However, some of these events could in fact be affected by pile up. The fraction of valid events caused by two

or more photons (either hitting the same pixel during one read-out cycle or producing a valid pattern that is incorrectly assigned to a single photon) is given by `NPVALID`. Similarly, `NINVALID` gives the total number of rejected events that were correctly classified as invalid during the simulation and `NPINVALI` gives the fraction of these invalid events affected by pile up.
Typically, the number of valid events `NVALID` is equal to the number of total events in the output event file. Events classified as invalid are discarded by default during the simulation, but can be included in the output file by setting the `sixtesim` parameter `SkipInvalids=no`.

Exercise: To get a better feeling for what is going on, plot the value of the FITS keywords `NVALID`, `NPVALID`, `NINVALID`, and `NPINVALI` as a function of the input flux. Why is the pile up fraction `NPVALID/NVALID` a good measure for the scientific quality of the data?

Exercise: Generate a SIMPUT file with a harder spectral shape (e.g., $\Gamma = 1.5$) than our example source but with the same flux. Place the source at a position that is $10''$ away from our source. Merge both SIMPUT files with the `simputmerge` tool (use `plist` to learn about the parameters of this tool!). Then run a 5 ks simulation and study how well you can separate both sources.
One way to do so is to make an image and use the histogramming function of `ds9`. Alternatively, use the FTOOL `fhisto` and bin the event file on the `RA` or `Dec` columns.

For the spectral analysis of off-axis sources you will need a corrected ARF that accounts for changes in the effective area (e.g, due to vignetting). SIXTE provides the tool `sixte_arfgen` to generate corrected ARFs for sources within a user-supplied region. The general calling syntax of the tool is

```
$SIXTE/bin/sixte_arfgen \
    ARFCorr=arf_corrected.fits \
    XMLFile=${xml} \
    RA=0.0 Dec=0.0 \
    RefRA=0.0 RefDec=0.0 \
    regfile=circle.reg
```

where `ARFcorr` specifies the name of the corrected ARF (the output file of the tool) and `XMLFile` gives the path to the XML-File that was used for the simulation. The parameters `RA`/`Dec` give the pointing of the observation, `SourceRA`/`SourceDec` specify the source position and `RefRA`/`RefDec` specify a reference position needed for the creation of a WCS system. The `regfile` parameter specifies the region on which the ARF correction should be applied.
An attitude file can be specified instead of the pointing via the `Attitude` parameter. Specific subsets of the period described by this attitude can be selected either via the optional `TSTART` and `exposure` parameters or via the `GTIfile` parameter, which accepts a FITS GTI file. If these parameters are not given, the complete attitude is sampled. As an example, the call could look like this if using an attitude file:

```
$SIXTE/bin/sixte_arfgen \
    ARFCorr=arf_corrected.fits \
    XMLFile=${xml} \
    RA=0.0 Dec=0.0 \
    RefRA=0.0 RefDec=0.0 \
    regfile=circle.reg \
    Attitude=attitude.fits \
    TSTART=0 exposure=10000
```

Exercise: Repeat the previous exercise for a source separation of 5′′ and extract spectra for all sources. Generate corrected ARFs for the off-axis sources by adjusting the parameters of the `sixte_arfgen` call above and use your favorite X-ray spectral analysis program to see how the spectral shape is affected.

Exercise: In order to quantify the mixing of photons from both sources further, we can take a look at the diagnostic information contained in the event file. Specifically, use the FTOOL `fselect` and, using the row selection syntax, generate a new event file for one of the two sources by selecting the events in the region around the source. Determine the fraction of photons from the other source that "contaminate" the selection region by using the information in the `SRC_ID` column of the event file. For each event, this column contains information about the source in which it originated, in form of the ID of that source in the SIMPUT file.

10.3 Time Variability

Many X-ray astronomical sources are time variable. The SIMPUT format allows to characterize the source variability in several ways:

- As a (energy dependent) light curve, specified through an ASCII file (parameter `LCFile`, which expects a file in which each row contains two numbers, the time and the flux at that time). Make sure to end the ASCII file with a new line.
- As a stochastic process defined through its power spectrum, an ASCII file where each row contains a frequency and the power at that frequency,
- As parameters of the power spectrum, which is defined by the sum of several Lorentzians and a zero-centered low frequency Quasi-Periodic Oscillation (QPO).

In addition the SIMPUT format also allows the specification of source variability through energy dependent pulse profiles.
The Time Variability extension can be used to describe the time dependence of the source flux but it can also be used to create a dynamic spectrum with the source i.e. spectral model of the source that varies with time. The `TIMING` extension for SIMPUT files can be used to define a spectrum at a given time or phase in a periodic source with the relative flux of the source for the respective spectrum. If this is done the `SRC_CAT` should not contain a `'NULL'` in the `Spectrum` column. Note that the timing extension can have any arbitrary name as long at is correctly referenced in the SIMPUT source catalogue. For simplicity, we will call it `TIMING` in the following, which is the default.

10.3.1 Light Curve

The `simputfile` call can be used to create a SIMPUT file with a light curve describing the time variability. For this a light curve in ASCII format needs to be created. The first column defines the time in seconds and the second column is the relativ flux. This flux is given as fraction of the `srcFlux`. The file then should look like this:

```
0.0           1.0
1.592568e+06  1.1710184667753536
4.746168e+06  1.3358780211173131
6.606792e+06  1.3293381514300098
         ...                 ...
```

An example light curve can be downloaded here: `https://www.sternwarte.uni-erlangen.de/~sixte/downloads/example_lightcurve.dat`
The `simputfile` call with the spectrum `mcrab.xcm` (see above) then looks like this:

```
#!/bin/bash

$SIXTE/bin/simputfile Simput=mcrab_lightcurve.fits \
                      Src_Name=first \
                      RA=0.0 \
                      Dec=0.0 \
                      srcFlux=2.137e-11 \
                      Elow=0.1 \
                      Eup=15 \
                      NBins=1000 \
                      logEgrid=yes \
                      Emin=2 \
                      Emax=10 \
                      MJDREF=55000 \
                      LCFile=example_lightcurve.dat \
                      XSPECFile=mcrab.xcm
```

As we are now dealing with time variable effects, we need to specify an absolute time reference. This is done by setting `MJDREF` to the desired value. **Important:** the simulation should be performed by the same `MJDREF` and the light curve has to cover the full range of the simulated observation.
Inspect the `mcrab_lightcurve.fits` file and have a look how the light curve is included in the source catalog. You will notice that the column `TIMING` is now referring to the light curve extension with `[TIMING,1]`.
Having a closer look at the `TIMING` extension we defined, reveals that the light curve is stored as scalar values, with each row containing one time and flux value.

```
  No. Type      EXTNAME      BITPIX Dimensions(columns)      PCOUNT  GCOUNT

   3  BINTABLE TIMING          8     12(2) 1001                  0    1

      Column Name                Format     Dims       Units     TLMIN  TLMAX
      1 TIME                       D                    s
      2 FLUX                       E
```

For a simulation of a particular source, the absolute flux $F_{\rm src}(t)$ at time t is determined by the reference value $F_{\rm src}$ as specified in the column `FLUX` of the source catalog and the light curve value $l(t)$ via the following relation:

$$F_{\rm src}(t) = F_{\rm src} \cdot l(t) \tag{7}$$

Note that the full specification of the `TIMING` extension of a SIMPUT file can be found in the definition document `http://hea-www.harvard.edu/heasarc/formats/simput-1.1.0.pdf`.
Exercise: Define a 1000 second light curve in a text file, linearly decreasing in flux to 0. Then create a SIMPUT file and simulate it again with `sixtesim`.
To produce a light curve you need to count the number of events in each time bin. This can either be done using the FTOOL `fhisto` (selecting events using the FITS file selection syntax), or using SIXTE's `makelc`-tool:

```
makelc EvtFile=sim_mcrab_lightcurve.fits \
       Lightcurve=sim_mcrab.lc \
       length=1000.0 \
       dt=1.0
```

This example call uses all event data to generate a light curve of 1000.0 s duration with a time resolution of `dt` = 1.0 s. You can use the FITS file selection syntax to select only the source events or a specific energy band (the latter can also be done using `makelc`'s `Emin` and `Emax` parameters). You can then use `fv` or the `fplot` FTOOL to plot the light curve:

```
computer:~/> fplot sim_mcrab.lc
Name of X Axis Parameter[error][x] -
Name of Y Axis Parameter[error] up to 8 allowed[y] counts
Lists of rows[-]
Device: /XWindow, /XTerm, /TK, /PS, etc[/xs]
Any legal PLT command[ ]
PLT> quit
```

Note that in the X axis, we simply plot the row number of the file – this is due to the fact that the output of `makelc` does not contain a dedicated `TIME` column. Instead, it provides header keys that map the row number to the time of the corresponding bin of the light curve, following conventions for equispaced binned light curves.
Exercise: Create a light curve from the event file and verify it decreases linearly to 0.
Exercise: The FITS extension syntax is very powerful. Read its documentation (`fhelp colfilter`). Extract the light curve again, but this time with a bin size of 2 s. Use `fplot` to plot the light curve as a count rate (i.e., introduce a virtual column in which you divide the counts by the bin size).

10.3.2 Periodic Variability

The SIMPUT format also allows to define a source with a periodic source variability. This works almost identically to defining a SIMPUT with a light curve. The important difference is that now the column in the `TIMING` extension is called `phase` instead of `time`. Additionally, the following header keywords have to be defined:

`PHASE0` phase of periodic oscillation at $t = 0$, which is defined by `MJDREF` and `TIMEZERO` (recommended value between 0 and 1)

`PERIOD` duration of one oscillation period (units given by `TIMEUNIT`)

An example of a SIMPUT with a periodic variability can be downloaded here: `https://www.sternwarte.uni-erlangen.de/~sixte/simput/crab.simput.tgz`
Exercise: Download the above SIMPUT of the Crab and compare it with the time variable SIMPUT you created in Sect. 10.3.1. Note that the light curve extension in the above file you just downloaded has a different name (which?). Can you spot all differences in the light curve extension between these two SIMPUT files?

10.3.3 Power Spectrum

To generate a time variable source with a Power spectrum, one can also use the `simputfile` call. Ingestigating it with `plist simputfile` you can see that there are many options to define QPOs (see A.2.1 for more information). For this add a QPO with a high Q-factor (e.g. 150 - 200) at a frequency of 0.01 Hz to the script that generated `mcrab.fits`. Then rerun the simulation and extract a light curve again.
Exercise: Use the FITS row filter options (`fhelp rowfilter`) to select only events that belong to our source and that are in the energy range between 0.5 keV and 1.0 keV.

10.3.4 Time Variable Spectra

The TIMING extension for SIMPUT files can also be used to define a spectrum at a given time or phase in a periodic source with the relative flux of the source for the respective spectrum. If this is done the SRC_CAT should not contain a 'NULL' in the Spectrum column. Note that the timing extension can have any arbitrary name as long at is correctly referenced in the SIMPUT source catalogue. For simplicity, we will continue calling it TIMING in the following.
This TIMING extension can contain a separate spectrum for each phase bin. The Phase and Relative Flux colums for the extensions containing the light curve should have one value for each spectrum in the extension. If there is only one spectrum being used but the relative flux varies then there should be multiple rows in the TIMING extension that call the same spectrum with a different relative flux. The flux of any defined spectra is defined by the value of the Flux column in the SRC_ID, which corresponds to a Relative Flux of 1.
In order for the multiple spectra defined to vary continuously between the specified time points, SIXTE linearly interpolates for the time intervals between the spectra. In the case of short periodicity in the source ($P \approx 10$ ks) this interpolation can mean you will rarely see any of the defined spectra without seeing elements of another.
An example of such a timing extension is shown below. The spectra need to be defined in a separate extension and referenced by the correct name. Therefore the relevant two extensions of the SIMPUT file could look like the following:

```
 No. Type      EXTNAME       BITPIX Dimensions(columns)       PCOUNT  GCOUNT

   2  BINTABLE SPECTRUM         8     8048(3) 2                    0    1

      Column Name                 Format     Dims       Units     TLMIN  TLMAX
      1 ENERGY                      1000E               keV
      2 FLUXDENSITY                 1000E               photon/s/cm**2/keV
      3 NAME                        48A

   3  BINTABLE TIMING       8     40(3) 2                         0    1

      Column Name                 Format     Dims       Units     TLMIN  TLMAX
      1 PHASE                       1E
      2 FLUX                        1E
      3 SPECTRUM                    32A
```

The TIMING extension itself simply contains in each row the phase, the relative flux, and the reference to the spectrum.

```
        PHASE            FLUX              SPECTRUM

      1   0.0000000E+00   1.0000000E+00   [SPECTRUM,1][#row==1]
      2   5.0000000E-01   1.0000000E+00   [SPECTRUM,1][#row==2]
```

10.4 Deep Field Simulations of the WFI

In the next step of the tutorial we will simulate and analyse deep field observations with the *Athena* WFI. We separate this into the generation of the simulation input (Sect. 10.4.1), which is fully independent of the chosen instrument, followed by an observation with all four chips of the WFI (Sect. 10.4.2). Of course, this second step can also be done with any other satellite and detector by simply replacing the chosen XML files.

10.4.1 SIMPUT for a Large Field

First, we will create a more complicated SIMPUT file consisting of several sources with different spectra. There are several ways to perform this task. The most direct is to construct the file with an external program, following the SIMPUT standard (see SIMPUT manual[31]).
Here we present a different way, constructing single sources with `simputfile` and merging those with the tool `simputmerge`. If a large number of images and spectra are used for several sources, this approach will become inefficient, but for smaller tasks it is faster than writing a dedicated program to generate SIMPUT files.

Creating a SIMPUT for several sources with `simputfile` We first construct SIMPUT files for a number of single point sources (see also Sect 10.2). We assume from this general introduction that we have a spectral file "spectrum.xcm", which holds our desired spectral information. Then we simply execute `simputfile` multiple times. Note that we use the `bash` syntax in order to store the information in a variable for reusing. It is best to put all of the following statements in a shell-script file to make it easier to write and edit the commands. See Sect. 10.2 and Sect. A for an explanation of the keywords for this tool.

```
#! /bin/bash
simpar="XSPECFile=spectrum.xcm Emin=0.5 Emax=10.0 clobber=yes"
simputfile RA=40.21 Dec=12.82 srcFlux=8.3e-12 Simput="src_00.fits" $simpar
simputfile RA=40.31 Dec=12.83 srcFlux=2.3e-11 Simput="src_01.fits" $simpar
simputfile RA=40.12 Dec=12.73 srcFlux=6.3e-12 Simput="src_02.fits" $simpar
simputfile RA=40.27 Dec=12.81 srcFlux=4.1e-12 Simput="src_03.fits" $simpar
simputfile RA=40.29 Dec=12.73 srcFlux=3.2e-11 Simput="src_04.fits" $simpar
simputfile RA=40.33 Dec=12.81 srcFlux=1.3e-11 Simput="src_05.fits" $simpar
```

Now that we have created the SIMPUT files for single sources, we merge them into a single file using `simputmerge`:

```
simputmerge \
  Infiles=src_00.fits,src_01.fits,src_02.fits,src_03.fits,src_04.fits,src_05.fits \
  Outfile=merged_simput.fits \
  clobber=yes FetchExtensions=yes
```

The keyword `FetchExtensions` is set to `yes`, in order to combine all available information in one file in the end. We can then also remove the initial and intermediate files with

```
rm src_*.fits
```

We have now created a very basic source catalogue called `merged_simput.fits`, which we can analyse with the WFI. You can also have a look at the SIXTE homepage and into the later sections of this tutorial for more complicated realistic examples.

10.4.2 Simple Wide Field Simulations

In the following we want to simulate an observation that uses all four chips of the WFI. This approach requires us to simulate four chips at once which can be done by providing an XML file to `sixtesim` which contains all four chips. We store the XML filename in a variable in order to increase readability of the tutorial. The following commands are again given for the `bash` shell.[32] In order to set up the configuration and make it easily accessible for the simulations later, we store the following information in a file called `setup.bash`:

[31] `http://hea-www.harvard.edu/heasarc/formats/simput-1.1.0.pdf`

[32] If you prefer, you can also give the complete path to the XML for each of the commands in the following, by simply replacing the symbol `$xml` with the full path to the respective XML file.

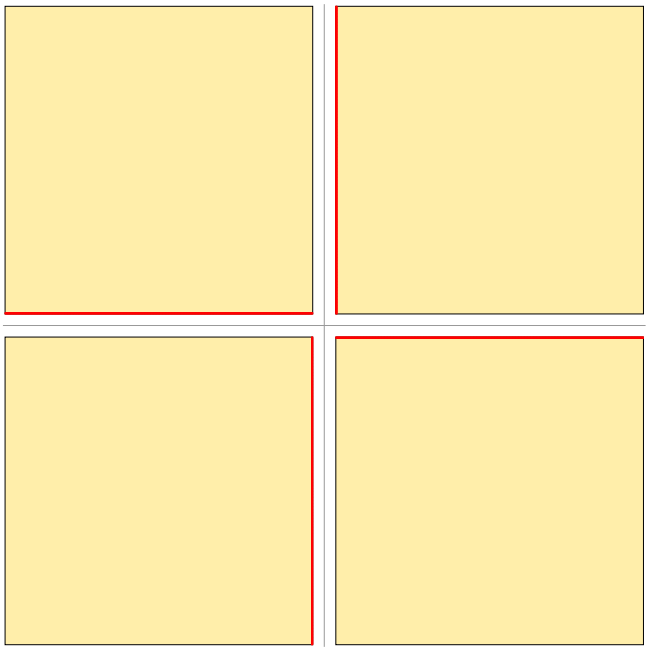

Figure 15: The WFI geometry as created by the tool `xml2svg`. The detector is shown in yellow. The aim point of the satellite is where the grey lines are crossing. In red a few pixels are drawn (not visible at this resolution due to the large number of pixels of the WFI). The pixels start where the readout begins and therefore indicate the readout direction. Hence, this means that, e.g., the upper left chip is read out from below.

```
#! /bin/bash

xmldir=$SIXTE/share/sixte/instruments/athena-wfi/wfi_wo_filter
xml=${xmldir}/ld_wfi_ff_all_chips.xml
```

Viewing the chip geometry with `xml2svg` **Warning: This feature has not yet been ported to SIXTE version 3!**
In order to get a feeling for the simulated geometry and for the aim point of the satellite, there is a SIXTE-tool called `xml2svg`. It can output an image of the detector geometry for any valid XML that follows the SIXTE standard (see also Appendix C). For the above XML files a call of this function would look as follows (you need to insert the full name for the XML to test it):

```
. setup.bash
xml2svg \
   XMLFiles="${xml}" \
   SVGWidth=3000 Border=10 SVGName=athenawfi_fullframe.svg
```

The parameters of the `SVGWidth` and `Border` are given in units of mm. The optional parameter of `DrawN` specifies how many single pixels (in red) are drawn. These pixels are drawn in the readout-direction. Figure 15 shows the resulting image. The SVG-file can be viewed in any WWW-browser or it can be converted to pdf using, e.g., the `convert`-program of the ImageMagick suite:

```
convert test.svg test.pdf
```

Simulating the four WFI chips with `sixtesim` Now that we understand the detector setup we can start running the simulation. The tool `sixtesim` is used for this purpose, as with any other instrument. We again put the command in a shell-script, sourcing first the setup we created before and then executing the simulation.

```
#! /bin/bash
```

```
. setup.bash

sixtesim \
  XMLFile=${xml} \
  RA=40.21 Dec=12.77 \
  Prefix=sim_ \
  Simput=merged_simput.fits \
  EvtFile=evt.fits \
  Exposure=1000 \
  clobber=yes
```

The output of the simulation is an event file for each of the chips, named in our case `sim_chip[0,1,2,3]_evt.fits`. You can specify the last part of the filename by the `EvtFile` keyword. Note that intermediate products such as the raw event file and impact list can be generated when giving the respective names as keywords (`RawData` and `ImpactList`). `RA` and `Dec` specify the pointing of the satellite in degrees. Make sure that these values point the satellite to a location that contains a source!
Using the FTOOL `ftmerge`, the event files can be easily merged into one larger event file. In most cases this is the most convenient way to proceed. The command in our case is

```
ftmerge \
  sim_chip0_evt.fits,sim_chip1_evt.fits,sim_chip2_evt.fits,sim_chip3_evt.fits \
  sim_combined_evt.fits clobber=yes
```

This call creates the combined output event file called `sim_combined_evt.fits`. You can now look at the structure of the event file with

```
fstruct sim_combined_evt.fits
```

Note the large number of events. Alternatively, take a closer look at the contents of the file with `fdump` or `fv`:

```
fv sim_combined_evt.fits
```

You can see each event is registered here. As this is a simulation, more than just the regular information from a real detector event file is available. Most notably, the column `PILEUP` tells you if this is a pile-up event, and the `SRC_ID` gives the information from which source the photon came (corresponds to the ID in the SIMPUT file). Note that in the combined event file the `RAWX/Y` coordinates are not meaningful, as they are defined on each of the chips. However, `RA` and `Dec` are still absolutely specifying the original position of origin of each event in the given list.

Creating an image with `imgev` Now we can use this file to create an image of full WFI detector. This can be done, e.g., with the SIXTE tool `imgev` with the following command and the resulting image is shown in Fig. 16.

```
imgev \
  EvtFile=sim_combined_evt.fits \
  Image=sim_combined_img.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=1063 NAXIS2=1063 CUNIT1=deg CUNIT2=deg \
  CRVAL1=40.21 CRVAL2=12.77 CRPIX1=532 CRPIX2=532 \
  CDELT1=-6.207043e-04 CDELT2=6.207043e-04 history=true \
  clobber=yes
```

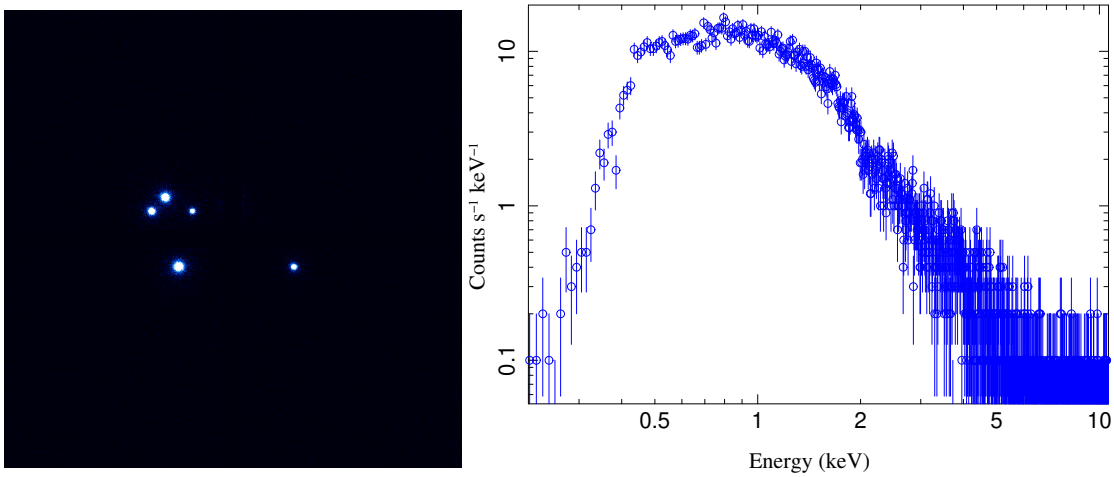

Figure 16: *Left:* The image of the four WFI chips created from the event file of the simple sample simulation (`sim_combined_img.fits`). Note that only 5 of the 6 sources are visible, as one fell into the gap between the chips. *Right:* Spectrum of the source located at lower, right, in the image, extracted from the same event file.

The input event file is given by the `EvtFile` parameter, which accepts the cleaned event file from any SIXTE simulation. The output file name is specified by `Image`. The other parameters specify the World Coordinate System (WCS) and bin the image to the desired pixel size. We choose `NAXIS1` and `NAXIS2` to accomodate the four 512 by 512 pixel chips of the detector, adding additional pixels to account for the gap between the chips. `CRVAL1` and `CRVAL2` give the aim point of the satellite, corresponding to RA, and Dec, respectively. In detector coordinates of the current setup this corresponds to the pixel coordinates `CRPIX1` and `CRPIX2` as given above. Finally, `CDELT1` and `CDELT2` are the pixel size in degrees (i.e., for the WFI, 130 μm, projected onto the sky using the 12 m focal length of the telescope). Note that the coordinate system is defined such that `CDELT1` has a negative increment. You can get additional information (as for every SIXTE tool) by using the `plist` command:

```
plist imgev
```

Important note: By simply changing the aim point of the satellite in the above command, it can be re-used for any WFI observation with the full four chips.

Creating a spectrum with `makespec` With the SIXTE tool `makespec` we can then also create a spectral file for one of the sources from our event file. Therefore, using `ds9` we have a look at the FITS image we produced above and select a region around one of the sources in RA and Dec. We will use the lower source at the right side in the following, which is centered at RA = 40.12° Dec = 12.73°.

```
makespec \
  EvtFile=sim_combined_evt.fits \
  Spectrum=sim_combined_spec.pha \
  EventFilter="RA>40.10 && RA<40.14 && Dec>12.71 && Dec<12.75" \
  RSPPath=${xmldir} clobber=yes
```

Besides the obvious names of the input (`EvtFile`) and output (`Spectrum`) filenames, also the path to the RMF and ARF have to be given with `RSPPath`, if these files are not in the current working directory. The most important field which needs to be given here is `EventFilter`. It makes use of the *extended filename syntax* of FITS, selecting events from a square box around the source position[33]. Alternatively, you can also generate a

[33] To get more information about the extended filename syntax, use the command `fhelp rowfilter`.

region file for the source using `ds9` and use the `regfilter` function of the FITS extended filename syntax. The resulting spectrum is shown in Fig. 16.

10.4.3 Simulating the *Chandra* Deep Field South

Generally, more complicated source catalogs will need to be assembled with scripts. One example is the *Chandra* Deep Field South (CDFS), which has been created by using the data provided by Lehmer et al. (2005) (`CDFS_cat_lehmer.fits`). Additionally, we will also use the background galaxy clusters as described by Finoguenov et al. (2015) (`CDFS_cat_galaxies.fits`). Both SIMPUT files are available from the SIXTE webpage at `https://www.sternwarte.uni-erlangen.de/~sixte/simput/CDFS_combined_simput.tgz`. Please download the file, save it in your simulation folder and extract it with

```
tar xfvz CDFS_combined_simput.tgz
```

Additional information: The CDFS data encoded in the SIMPUT have been directly verified with the official Chandra simulator (MARX[34]), which as of its version 5.3 can also read SIMPUT files.
Now we can run the simulation with a similar command as before. The only difference is that we have two SIMPUT files, which we specify as a comma-separated list of filenames. Additionally, we choose a slightly larger exposure of 5 ksec.

```
sixtesim \
 RA=53.13 Dec=-27.8 \
 Prefix=cdfs_ EvtFile=evt.fits \
 XMLFile=${xml} \
 Simput=CDFS_cat_lehmer.fits,CDFS_cat_galaxies.fits \
 Exposure=5000 \
 clobber=yes
```

After we have combined the events into the combined event file with

```
ftmerge \
  cdfs_chip0_evt.fits,cdfs_chip1_evt.fits,cdfs_chip2_evt.fits,cdfs_chip3_evt.fits \
  cdfs_combined_evt.fits clobber=yes
```

the image can now again be created by a similar command as before:

```
imgev \
  EvtFile=cdfs_combined_evt.fits \
  Image=cdfs_combined_img.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=1063 NAXIS2=1063 CUNIT1=deg CUNIT2=deg \
  CRVAL1=53.13 CRVAL2=-27.8 CRPIX1=532 CRPIX2=532 \
  CDELT1=-6.207043e-04 CDELT2=6.207043e-04 history=true \
  clobber=yes
```

Looking at the image file with `ds9` will yield something similar to Fig. 17. Due to the fixed pointing of the satellite the four chips and the gaps inbetween are directly visible.

[34] `http://space.mit.edu/cxc/marx/`

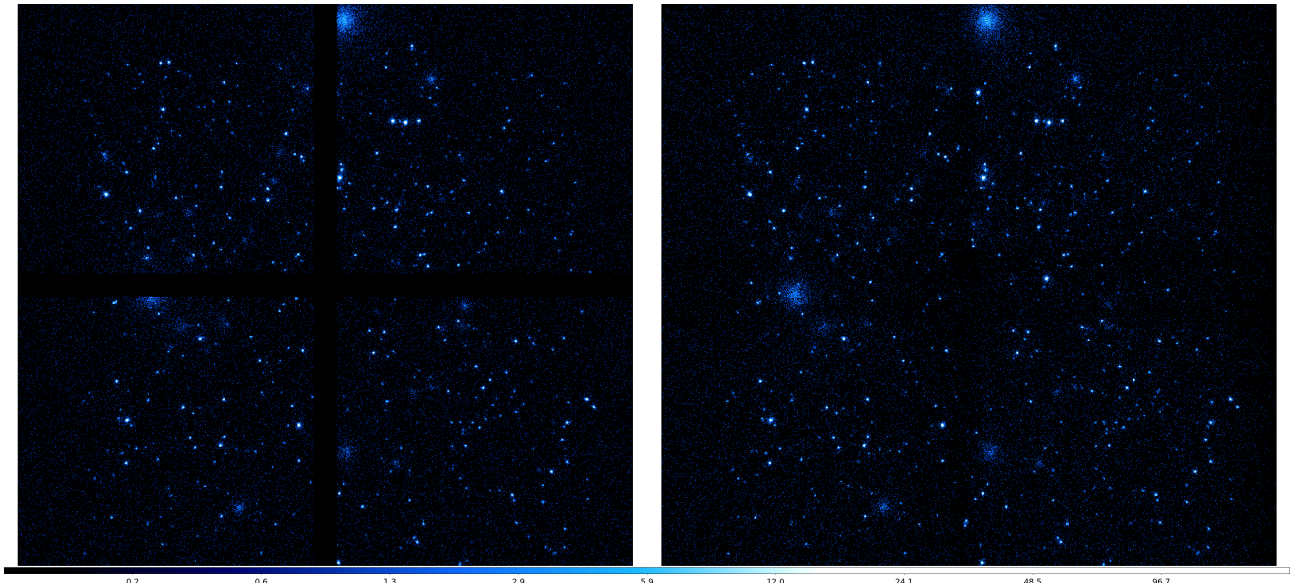

Figure 17: *Left:* Image of the 5 ksec simulation of the CDFS. The bright point sources can be seen, as well as the larger and fainter galaxy clusters in the background. The four chips and their gaps are clearly visible. *Right:* Same exposure and field of the sky, but now including a Lissajous pattern as attitude file (see Fig. 18). Note that this image is not exposure corrected, but shows directly the measured counts.

10.4.4 Dithering and Exposure Map

In order to avoid to imprint the gaps between the chips in the data (see Fig. 17), the satellite will usually be operated in a dithering mode. In SIXTE, a time-varying pointing of the satellite can be specified with an attitude file. The attitude file is a FITS file which for a given `Time` (in seconds) contains the position of the satellite `RA` and `Dec`. Additionally the rotation of the satellite can be specified by the additional column `beta_sat` (it will rotate the chip in the observer plane). Such an attitude file can be generated manually or with the `attgen_dither` SIXTE tool. An example call for the `attgen_dither` tool, tailored to the CDFS simulation, can be seen below:

```
attgen_dither \
  Attitude=attitude_lissajous.fits \
  Amplitude=0.035 \
  SrcRA=53.13 \
  SrcDec=-27.8 \
  Exposure=5000
```

This call to the `attgen_dither` tool creates a Lissajous dithering pattern with a given amplitude (in degrees), centered around `SrcRA` and `SrcDEC` (in degrees). The time interval in seconds covered by the attitude file can be specified with the Exposure parameter. The optional parameters `TSTART` and `MJDREF` adjust the starting time and the reference Modified Julian Date respectively. The number of steps for the pattern can be specified with `nbins`. The left panel of Fig. 18 shows the satellite's pointing as specified by this attitude file.
Using this attitude file, we can then also calculate the *Exposure Map*. This map specifies the amount of time for which each part of the sky has been observed by the satellite. The relevant SIXTE tool is called `exposure_map`. To calculate a short 5 ks snapshot of the attitude at a lower time resolution, we can use the following call:

```
exposure_map \
```

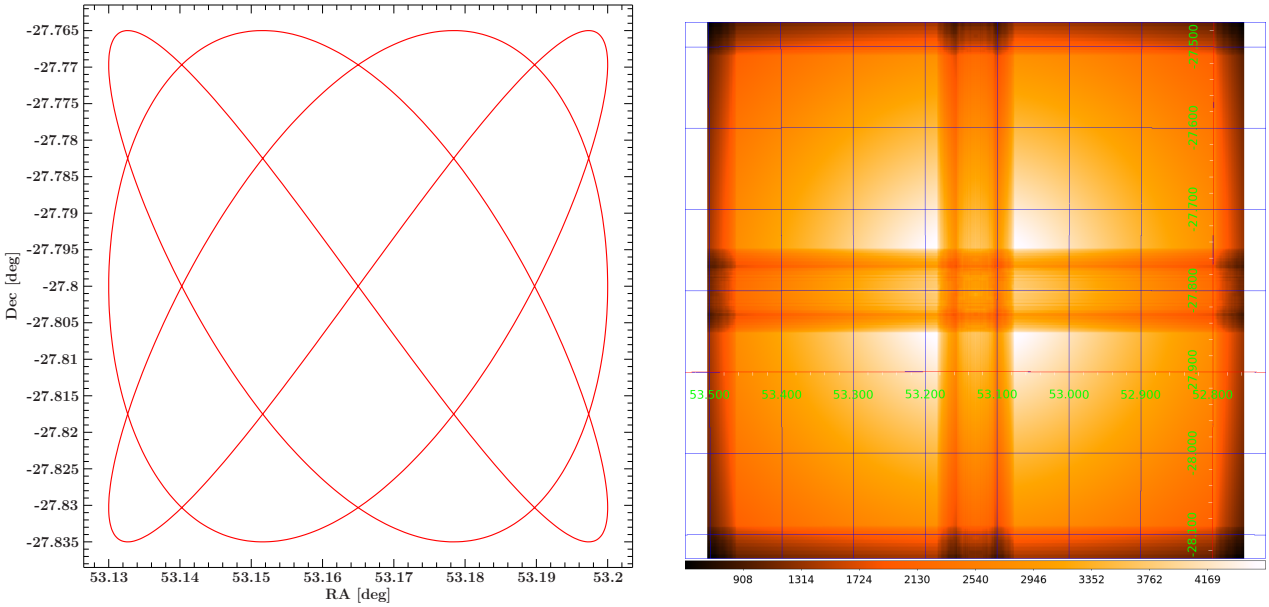

Figure 18: *Left:* The pointing of the satellite in RA and Dec as specified by the attitude file generated using `attgen_dither` tool. *Right:* 5 ksec exposure map calculated with the `exposure_map` tool. The four chips are clearly visible.

```
Vignetting=${xmldir}/athena_vig_13rows_20231211.fits \
Attitude=attitude_lissajous.fits \
Exposuremap=expo_map.fits \
XMLFile=${xml} \
fov_diameter=70 \
CoordinateSystem=0 projection_type=TAN \
NAXIS1=1063 NAXIS2=1063 CUNIT1=deg CUNIT2=deg \
CRVAL1=53.13 CRVAL2=-27.8 CRPIX1=532 CRPIX2=532 \
CDELT1=-6.207043e-04 CDELT2=6.207043e-04 \
TSTART=0 timespan=5000.000000 dt=100. \
chatter=3 clobber=true
```

The right panel of Fig. 18 shows the resulting exposure map, viewed with `ds9`. Note that we need to also specify the vignetting file (`Vignetting` keyword), as the flux from sources at the outer regions of the field of view is lower than in the central parts. We specify the attitude file with `Attitude` and the filename of the output map is set by `Exposuremap`. The exposure map calculation starts at `TSTART` with time increment `dt` and has a duration given by `timespan` (in seconds). The grid on which the exposure map is calculated is specified via WCS keyrecords, similar to the `imgev` calls in the previous sections. The `projection_type`-argument defines the map projection that is to be used. The `NAXIS1` and `NAXIS2` keywords give the number of x- and y-pixels in the calculated exposure map. `CRVAL1` and `CRVAL2` define the center of the map (i.e., the pointing of the observation in this example) and `CRPIX1` and `CRPIX2` define the corresponding pixel coordinates. Finally, `CDELT1` and `CDELT2` give the pixel sizes in degrees. Note that the coordinate system is defined such that `CDELT1` has a negative increment.

The calculation time scales linearly with the number of pixels contained in the resulting image. Additionally the keyword `fov_diameter` (in arcmin) should be set. While the exact geometry and size of the FOV is set in the XML, setting this to a conservative value and overestimating the actual value greatly helps the speed of the

simulation, as everything beyond this value will be completely ignored. Therefore we set it to 70′ for the WFI detector in this case.
Now we are able to start the full WFI simulation with the given attitude file in order to remove the gaps between the chips. Very similar to above, the commands to obtain the image then are as follows (note that `RA` and `Dec` do not need to be given since they will internally be overwritten by the correct attitude):

```
sixtesim \
  Attitude=attitude_lissajous.fits \
  Prefix=c_att_ \
  XMLFile=${xml} \
  Simput=CDFS_cat_lehmer.fits,CDFS_cat_galaxies.fits \
  Exposure=5000 \
  clobber=yes

ftmerge \
  c_att_chip0_evt.fits,c_att_chip1_evt.fits,c_att_chip2_evt.fits,c_att_chip3_evt.fits \
  c_att_combined_evt.fits clobber=yes

imgev \
  EvtFile=c_att_combined_evt.fits Image=c_att_combined_img.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=1063 NAXIS2=1063 CUNIT1=deg CUNIT2=deg \
  CRVAL1=53.13 CRVAL2=-27.8 CRPIX1=532 CRPIX2=532 \
  CDELT1=-6.207043e-04 CDELT2=6.207043e-04 history=true \
  clobber=yes
```

Now we produced an image without the gaps, by slightly changing the attitude of the satellite (`c_att_combined_img.fits`). The image is shown in the right panel of Fig. 17. Note that it is not exposure corrected, but directly shows the counts as seen by the WFI.

10.4.5 The next steps

Exposure correction The most important step missing above is the exposure correction of the final CDFS simulation. Simulating the full 80 ksec observation and an exposure map for the same time at a finer grid will yield the necessary files to perform this correction. Then, using an external analysis software of your choice (`python`, `ISIS`, `IDL`, ...) both files need to be loaded with their respective WCS coordinates. For each pixel in the final image you can then look up the exposure from the map and correct for it.

Adding background Additional background (such as cosmic X-ray background) can be easily added with an extra SIMPUT file. A selection of such files can be found at the SIMPUT section of the SIXTE homepage.

10.4.6 Used SIXTE tools in this section

The following SIXTE tools were used and explained in this section:

- `simputfile`
- `simputmerge`
- `xml2svg`
- `attgen_dither`
- `sixtesim`

- `imgev`
- `makespec`
- `exposure_map`

10.5 Extended source simulations

So far, we have only dealt with simulations for single or multiple point sources. In the next step of the tutorial we give an example for simulating data from extended sources.

10.5.1 Generating the SIMPUT file

The input for simulations of extended sources are source images which describe the spatial distribution of the flux. Such images can be obtained, e.g., from earlier observations of a source or from models for the emission of an extended source (e.g., in N-body simulations).
If using observational data, be aware that features caused by the instrument that made these images will also be present in the simulated data. Such features will be, e.g., "spots" or photon noise in low signal to noise regions of the image (these can often be compensated for by convolving the original image with a Gaussian filter, albeit at a loss of resolution), effects of the PSF of the imaging instrument, or gaps such as those between the CCDs of *XMM-Newton*. Typically, for the simulation not to be influenced by these effects you want to choose an image that has a high signal to noise and that has a higher spatial resolution than that of the instrument for which the simulation is performed. Note that the old adage "garbage in – garbage out" also applies to simulations…
Here, we use a set of images obtained with *Chandra* and available from `http://chandra.harvard.edu/photo/openFITS/xray_data.html`. These images are already preprocessed, they are convolved with Gaussians to remove photon artifacts. For a large number of sources, images are available in several energy bands. For the example here, we use data from the colliding wind binary η Car, published by Hamaguchi et al. (2014). The *Chandra* web page contains images in three energy bands, 0.5–1.2 keV, 1.2–2.0 keV, and 2.0–7.0 keV.
In order to generate the SIMPUT file, we need to know the source position, the spectral shape, as well as the source flux in each of these bands. The image we just downloaded is the average of all ACIS observations during the X-ray minimum between 2003 and 2009. The source position can be obtained from the image files:[35]

```
computer:> fkeyprint etaCar_xray_hi.fits CRVAL1

# FILE: etaCar_xray_hi.fits
# KEYNAME: CRVAL1

# EXTENSION:    0
CRVAL1  =     161.267156643662
computer:> fkeyprint etaCar_xray_hi.fits CRVAL2

# FILE: etaCar_xray_hi.fits
# KEYNAME: CRVAL2

# EXTENSION:    0
CRVAL2  =     -59.684372315062
```

Using Hamaguchi et al. (2014, Fig. 2), we estimate the 3–8 keV flux of the central point source to 10^{-11} erg cm^{-2} s^{-1}. The diffuse emission also present in the images has an absorbed 0.2–10 keV flux of

[35] Looking at the file header, the `CRPIX1` and `CRPIX2` values show that the reference pixel is located in the center of the images and therefore on the source.

$2.2 \times 10^{-12}\,\mathrm{erg\,cm^{-2}\,s^{-1}}$ (Seward et al., 2001), we will ignore the small error we make and assign the central point source flux to the whole image.
Since Hamaguchi et al. (2014) do not give the spectral parameters for the observation, we describe the spectral shape by a constant in flux:

```
hde226868:~/projects/sixte.exercise/color> xspec

                XSPEC version: 12.9.0n
        Build Date/Time: Tue Jul 26 17:10:00 2016

XSPEC12>model power

Input parameter value, delta, min, bot, top, and max values for ...
              1       0.01(      0.01)         -3         -2          9         10
1:powerlaw:PhoIndex>1
              1       0.01(      0.01)          0          0      1e+20      1e+24
2:powerlaw:norm>1

========================================================================
Model powerlaw<1> Source No.: 1   Active/Off
Model Model Component  Parameter  Unit     Value
 par  comp
   1    1   powerlaw   PhoIndex            1.00000      +/-  0.0
   2    1   powerlaw   norm                1.00000      +/-  0.0
------------------------------------------------------------------------

XSPEC12>save model constflux
XSPEC12>quit
```

Again, in a real simulation for the source, one would do a deeper literature search and then assign a proper spectral shape to the image.
We can now build the SIMPUT file using the following script:

```
#!/bin/bash

RA=161.267156643662
Dec=-59.684372315062

simputfile Simput="etacar_high.fits" \
  RA=${RA} \
  Dec=${Dec} \
  srcFlux=1e-11 \
  Emin=3. \
  Emax=8. \
  Elow=2. \
  Eup=10. \
  XSPECFile=constflux \
  ImageFile=etaCar_xray_hi.fits \
  clobber=yes
```

The parameters `Elow` and `Eup` define the energy band for which the source spectrum is to be defined. Since `simputfile` requires `Elow` ≤ `Emin` < `Emax` ≤ `Eup`, and we only know the 3–8 keV flux, we set them to the same values.

10.5.2 Simulating the observation

To see how the hard X-ray image of η Car would look like in the WFI, we can now use the same tools that were already discussed in Sect. 10.4. To simulate a 1 ks observation of η Car, use the following script:

```
#! /bin/bash

xmldir=$SIXTE/share/sixte/instruments/athena-wfi/wfi_wo_filter

xml=${xmldir}/ld_wfi_ff_all_chips.xml

RA=161.56
Dec=-59.52

${SIXTE}/bin/sixtesim \
  XMLFile=${xml} \
  RA=${RA} Dec=${Dec} \
  Prefix=high_ EvtFile=evt.fits \
  Simput=etacar_high.fits \
  Exposure=1000 \
  clobber=yes
```

Note the warning that the spectrum is not defined for the whole energy band. RA and Dec are adjusted to avoid the gap between the chips of the WFI.
You can then look at the event files using `ds9`. The source is located on chip0. Note that you have to bin in RAWX and RAWY coordinations (in `Bin→Binning Parameters`[36]).
Exercise: Generate a spectrum of chip0 and verify that indeed only photons between 2 and 10 keV are generated. Where *do* the photons detected below 2 keV come from?
Exercise: Generate two more SIMPUT files using the medium and low band, merge them with `simputmerge`, and perform a new WFI simulation.
Exercise (harder): Rather than describing the spectral shapes with constants, describe the overall spectral shape of η Car and the Humunculus nebula using an *apec* model with a temperature of $kT = 4$ keV, absorbed by $N_{\rm H} = 2 \times 10^{22}$ cm^{-2}. Use XSPEC to determine the relative fluxes for the three energy bands for which *Chandra* images are available. Then generate three SIMPUT files, fixing the hard band flux to 10^{-11} cgs and the two lower fluxes to values appropriate for the absorbed *apec* model. Do not forget to set `Eup` and `Elow` appropriately. Perform the simulation again.
Exercise (even harder): For each of the three images, use the *Chandra* `dmcopy` tool and an appropriately defined region filter[37] to produce an image file of the humunculus nebula that does not include the central point source and an image file that only includes the point source. Prepare SIMPUT files that model the nebula with a different spectral shape than the point source, merge them with `simputmerge`, and run the simulation again, now having a very realistic model for the whole source.

[36] make sure you check the box "`or center of data`" as well
[37] `http://cxc.harvard.edu/ciao/ahelp/dmfiltering.html`

10.6 Observations with *eROSITA*

The general SIXTE simulation for *eROSITA* requires the simulation of seven telescopes simultaneously. This can also be done with sixtesim, by specifying XML files for all seven telescopes to a single call.
In case only a simulation of a single telescope is desired, it can be done as described in Sect. 10.2, and by choosing one of the telescope XMLs (e.g., `${xmldir}/erosita_1.xml` for telescope 1), where we assume that the XML detector files are stored in the directory defined by `${xmldir}`.
If you want perform your own *eROSITA* simulations, especially if you desire to go beyond what shown below, have a look at the *eROSITA* specific section Sect. 6.3. Additional *eROSITA* tools on how to calculate a GTI file or an exposure map are briefly explained there.

10.6.1 Pointed Observations

A tutorial for performing pointed observations with SIXTE in general was already provided above, so we will only point out some key aspects and differences to *eROSITA* simulations here. Note that we assume that you have created a SIMPUT file called `crab.fits` with a source centered at RA=0.0 and Dec=0.0. Either you have constructed it yourself (see Sect. 10.2) or you can also download it from `https://www.sternwarte.uni-erlangen.de/~sixte/simput/athenacrab_flux0001000muCrab.simput.tgz`.
Since *eROSITA* has seven cameras, in contrast to a normal `runsixt` simulation call, seven XML files have to specified as a comma-separted list in the `XMLFile` argument:

```
sixtesim Prefix=crab_ \
       Simput="crab.fits" \
       XMLFile=${xmldir}/erosita_1.xml,${xmldir}/erosita_2.xml,${xmldir}/erosita_3.xml,${xmldi
       MJDREF=51543.8975 \
       Exposure=1000. \
       Ra=0 \
       Dec=0 \
       EvtFile=evt.fits \
       clobber=yes
```

This simulation run will produce seven files, named `crab_tel1_evt.fits` to `crab_tel7_evt.fits`. These contain the event data for each telescope.
The `MJDREF` parameter has been set to 51543.8975, which is the `MJDREF` assumed by all other *eROSITA* related tools in SIXTE, as well as all tools in the *eROSITA* Science Analysis Software System (eSASS) [38].
A caveat: If you want to start the simulation at a time different to the reference date given by the `MJDREF` keyword, which for *eROSITA* is 51543.8975, then this parameter has to be equal in all involved parts of the simulation, including TIMING extensions in the SIMPUT file.
We can create an image from this simulation by using the `imgev` tool with the appropriate pixel size and number for *eROSITA*. This can be done either after all event files are combined to one file (with `ftmerge` for example, see above on p. 58), or the image of a single telescope can be constructed:

```
imgev \
      EvtFile=crab_tel1_evt.fits \
      Image=image.fits \
      CoordinateSystem=0 Projection=TAN \
      CUNIT1=deg CUNIT2=deg \
      NAXIS1=384 NAXIS2=384 \
```

[38] `https://erosita.mpe.mpg.de/dr1/eSASS4DR1/`

```
CRVAL1=0.0 \
CRVAL2=0.0 \
CDELT1=-0.0027778 CDELT2=0.00277778 \
CRPIX1=192.5 CRPIX2=192.5 \
clobber=yes
```

With this command the image will be contained in the file `image.fits`. Note that `CRVAL1/2` has to be changed appropriately to the position of interest in the sky, i.e., the satellite pointing position for a pointed observation.

10.6.2 All-Sky survey

To perform an all-sky survey simulation with *eROSITA*, a survey strategy is needed, i.e., information about how the telescope will slew over the sky will have to be provided. This information is provided to SIXTE through so-called attitude files. These hold information of the pointing of the telescope at given times.

```
sixtesim Prefix=crab_ \
        Simput="crab.fits" \
        XMLFile=${xmldir}/erosita_1.xml,${xmldir}/erosita_2.xml,${xmldir}/erosita_3.xml,${xmld
        Exposure=10000. \
        Ra=0 Dec=0 \
        Attitude=attitude.fits \
        TSTART=0.0
        MJDREF=51543.875 \
        EvtFile=evt.fits \
        CLOBBER=yes
```

The attitude-file is specified with the `Attitude` parameter. It is a simple FITS file, which needs to provide the columns with the exact name `TIME`, `RA` (or `VIEWRA`), `DEC` (or `VIEWDEC`), and `ROLLANG`. If the roll angle is not given it will be set to zero by default. Recommended units for the time are seconds and degrees for the coordinates. Other units can also be used when specified accordingly in the FITS header.
Although the `RA` and `Dec` coordinates still have to be provided when calling `sixtesim`, they are overridden by the `Attitude` parameter and are not used in the simulation. Make sure that your attitude's `MJDREF` matches the one provided to the simulation and that the Attitude-file spans the entire time interval to be simulated. `TSTART` defines the start date of the simulation.

10.6.3 Simulating single sources in the All-Sky survey

If you want to simulated only part of the sky in the scanning mode as specified by the eROSITA all-sky attitude file, SIXTE offers an easy way to do so. While it is possible to change the attitude file, this approach is not recommended. The proper solution is to calculate a GTI file, which can be given to the `sixtesim` simulation such that only during these times the simulation is performed.
The SIXTE tool `ero_vis` calculates such a GTI file. It calculates from the given SIMPUT file, when the sources defined in this file are visible. It contains all time intervals in which the telescope slews over the predefined regions in the SIMPUT catalog for a provided attitude file.

```
ero_vis GTIFile=crab.gti \
    Simput="crab.fits" \
    Exposure=10000. \
    RA=0 \
```

```
Dec=0 \
Attitude=attitude.fits \
TSTART=0.0 \
dt=1.0 \
visibility_range=1.0 \
clobber=yes
```

The exposure parameter corresponds to the overall exposure of the survey. The attitude file is specified with the `Attitude` parameter and provides information about the scanning path of the telescope. The `RA` and `Dec` coordinates are overwritten by the predefined coordinates of the SIMPUT catalog. `TSTART` gives the starting point in time for the survey and `dt` defines the time steps in the simulation. The parameter `visibility_range` specifies the FoV in degrees. For eROSITA it is best kept at 1 deg. it is important that this parameter is at least as large as the actual FoV. The hereby obtained GTI file can be simply introduced as `GTIFile` parameter in the `sixtesim` call for the all-sky survey and thus, yields an observation in survey mode for selected regions and fields. If the SIMPUT contains extended images, those are also correctly taken into account.

10.6.4 Downloading eROSITA files

The most recent version of the eROSITA XML files can be downloaded from the SIXTE homepage (`https://www.sternwarte.uni-erlangen.de/sixte/instruments/`). The setup there already includes two different types of ARFs for the different filter combination of the telescopes, the correct rotation of the camera, and the measured PSF with off-axis angles.
The current attitude file for eROSITA can either be downloaded from the SIXTE homepage or the eROSITA wiki page (`https://wiki.mpe.mpg.de/eRosita/ScienceRelatedStuff`).
The following points are work in progress.

- photons from single reflection and sources outside the field of view are not yet included
- a simplified RMF is used for the simulation (the measured RMF can not be used, as it includes the lower energy threshold, which is directly); therefore smaller differences at lower energies are expected when trying to recover the input spectra after a simulation.

10.7 Simulations of Galaxy Clusters with the X-IFU

This section contains a tutorial focused on the simulation of observations of Galaxy Clusters with the *Athena* X-IFU. Even though we chose a specific type of sources, the simulation approach can be generalized to any extended source featuring significant spectral variation across the field of view. All files necessary to run the following simulations are available for download at `https://www.sternwarte.uni-erlangen.de/~sixte/downloads/X-IFU_clusters_tutorial.tgz`.

10.7.1 SIMPUT file for 3D data

The main difficulty of simulating a galaxy cluster observation lies in the building of a suitable SIMPUT file containing all the information probed by a high-resolution integrated field unit like the X-IFU. To illustrate this, let us first take a simple point source defined by an XSPEC spectrum with significant line emission (xifu_point_source.xcm file):

```
model  phabs*apec
          0.03     -0.001          0          0     100000       1e+06
           1.5       0.01      0.008      0.008         64          64
```

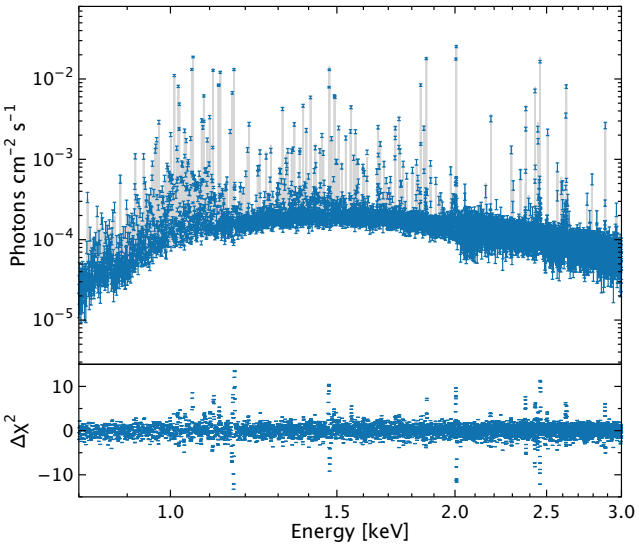

Figure 19: Illustration of the artifacts present around emission lines in case the input spectrum is not properly sampled when building the SIMPUT file. This figure was obtained with XSPEC after fitting the unbinned spectrum.

```
   0.75      0.001          0          0          5          5
      0      -0.01     -0.999     -0.999         10         10
      1       0.01          0          0      1e+20      1e+24
```

To obtain the corresponding SIMPUT file, one can for instance run the following command:

```
simputfile XSPECFile=xifu_point_source.xcm \
  RA=239.064583333 \
  Dec=66.3470277776 \
  srcFlux=2.e-12 \
  Emin=0.1 Emax=10 \
  Simput=xifu_point_source.simput
```

If one then analyzes the output spectrum of an X-IFU simulation performed with this SIMPUT file (see Sect. 10.7.2 for the simulation run and the `makespec` tool presented in Sect. 10.2 for the spectrum building), one will notice that significant artifacts are present near the emission lines (Figure 19). This is actually due to the default discretization of the spectra in the SIMPUT file being too raw, resulting in poorly defined lines. While it is suitable for most simulations, high resolution instruments like the X-IFU will reveal these artifacts. It is therefore necessary to correct this by using a finer sampling by adding `Estep=0.00025` to the `simputfile` call. This seems like a simple enough fix, but the resulting SIMPUT file will be ~ 2 Mbytes large while the previous one was only ~ 100 kbytes. If we extend this to at least one source per X-IFU pixel, one quickly realizes that the simulation can no longer be run on a standard computer.
In this section, we will thus present two different approaches to solve this issue: the first one will concentrate on the use of existing 2D spectral parameters maps obtained by other instruments to compare with the future X-IFU capability; the second one will present a general tool to interface SIXTE with galaxy cluster simulations. On top of those two solutions, the SIMPUT format (see SIMPUT manual) allows the definition of a source through a perfect list of photons (i.e. before being folded through any instrumental response). This approach should be chosen for the simulation of faint sources with high spatial variability for which too many different spectral shapes would need to be saved in the SIMPUT catalog while only a limited number of photons would be detected during a single observation.

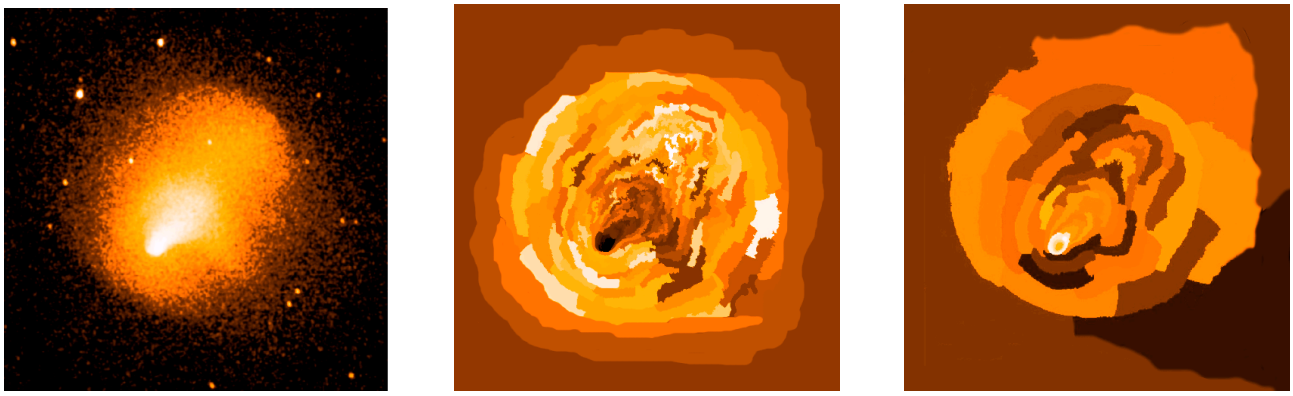

Figure 20: Parameter maps obtained by Russell et al. (2012) from *Chandra* observations on luminosity (left), temperature (center), and abundance (right)

.

Using existing 2D maps

To illustrate this approach, we will use the results obtained by Russell et al. (2012) with *Chandra* on the galaxy cluster Abell 2146 for which they could derive temperature and abundance maps (see Figure 20). The brute force approach would be to define a SIMPUT catalog containing one source per input image pixel. If this would be suitable for parameters randomly distributed across the images, astrophysical sources typically feature strong correlations between physical quantities. To take advantage of this property and reduce the amount of sources and spectra in the catalog, the `simputmultispec` tool was developed. It takes as input a series of parameter maps and will create a SIMPUT catalog containing extended sources corresponding to regions with similar parameters according to a given binning. We put below an example call:

```
simputmultispec Simput=clusterA2146.simput \
   XSPECFile=xifu_point_source.xcm \
   ImageFile="A2146_SXB_russel_coord_cal.fits" \
   ParamFiles="A2146_kt_russel_coord_cal.fits;A2146_ab_russel_coord_cal.fits" \
   ParamNames="2;3" \
   ParamsLogScale="yes;no" \
   ParamsNumValues="8;8" \
   Emin=0.5 Emax=10.0 \
   srcFlux=7.995076796356145e-12 \
   RA=239.064583333 Dec=66.3470277776 \
   Elow=0.2 Eup=12 Estep=0.00025
```

An X-IFU simulation can then be run by using the `sixtesim` tool and processed using the tools available in SIXTE (see 10.7.2) as illustrated in Figure 21. We can clearly see in it the changing spectral shape of the emission across the field of view from a single 100 ks observation with very high signal to noise. If one looks closely to the spectrum on the left, we can also notice the effect of the poorer spatial resolution of the X-IFU from the high energy excess due to the component mixing in the brightest parts of the cluster.
We note that this approach is especially suited for the comparison of the X-IFU capability with existing observations or for simulations for which the emission mixing along the line of sight is not a priority and a 2D description of the source is sufficient.

Using galaxy cluster simulations

In some cases, it is necessary to properly take into account the 3D character of a galaxy cluster emission and the

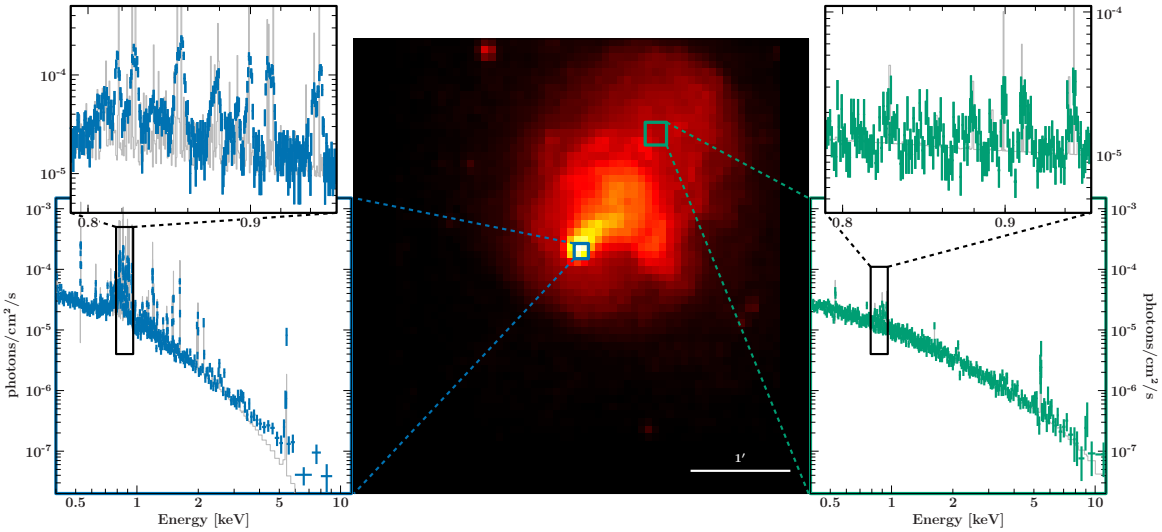

Figure 21: Illustration of a 100 ks X-IFU simulation of the Abell 2146 galaxy cluster. One can clearly see the spectral changes across the field of view as well as the wealth of information contained in the spectrum of a few X-IFU pixels.

previous solution is not well suited. A SIMPUT tool called `simputmulticell` was specifically designed to provide an interface between SIXTE and 3D data extracted from either toy models or cosmological simulations. To illustrate this approach we will use a simple cluster toy model whose density, and iron abundance depend on the radius with the following relations:

$$n_e(r) = n_0 \left[1 + \left(\frac{r}{r_n}\right)^2\right]^{-1} \tag{8}$$

$$Z_{Fe}(r) = Z_0 \left[1 + \left(\frac{r}{r_Z}\right)^2\right]^{-1/2} \tag{9}$$

with $n_0 = 0.015\,\mathrm{cm}^{-3}$, Z_0=0.7, $r_c = 100\,\mathrm{kpc}$ $r_Z = 200\,\mathrm{kpc}$. In order to illustrate the line-of-sight mixing in such simulations, we used the model proposed by Vikhlinin et al. (2006) and chose parameters leading to a significant difference between the cool core and maximal temperature in the simulation:

$$T(r) = T_0 t_{\mathrm{cool}}(r) t(r) \tag{10}$$

$$t_{\mathrm{cool}}(r) = \frac{x + T_{\mathrm{min}}/T_0}{x + 1} \tag{11}$$

$$x = (r/r_{\mathrm{cool}})^{a_{\mathrm{cool}}} \tag{12}$$

$$t(r) = \frac{(r/r_t)^a}{\left[1 + (r/r_t)^b\right]^{c/b}} \tag{13}$$

with $kT_0 = 6\,\mathrm{keV}$, $kT_{\mathrm{min}} = 1.8\,\mathrm{keV}$, $r_{\mathrm{cool}} = 40\,\mathrm{kpc}$, $a_{\mathrm{cool}} = 2$, $r_c = 860\,\mathrm{kpc}$, $a = 0.04$, $b = 3$, $c = 4.5$. The cluster is placed at redshift 0.1, such that 100 kpc corresponds to $\sim 1.5'$, i.e. well inside the X-IFU field of view. Using an external script, we construct from this toy model a 3D dataset saved in a FITS table containing for each point of the grid its position on the sky, temperature, iron abundance and X-ray flux (`xifu_3D_grid.fits`). We note that the box is 3 times larger in the line of sight direction compared to the X-IFU field of view. It is not necessary to include anything outside the X-IFU field of view in this dataset. But since we are dealing with a 3D structure

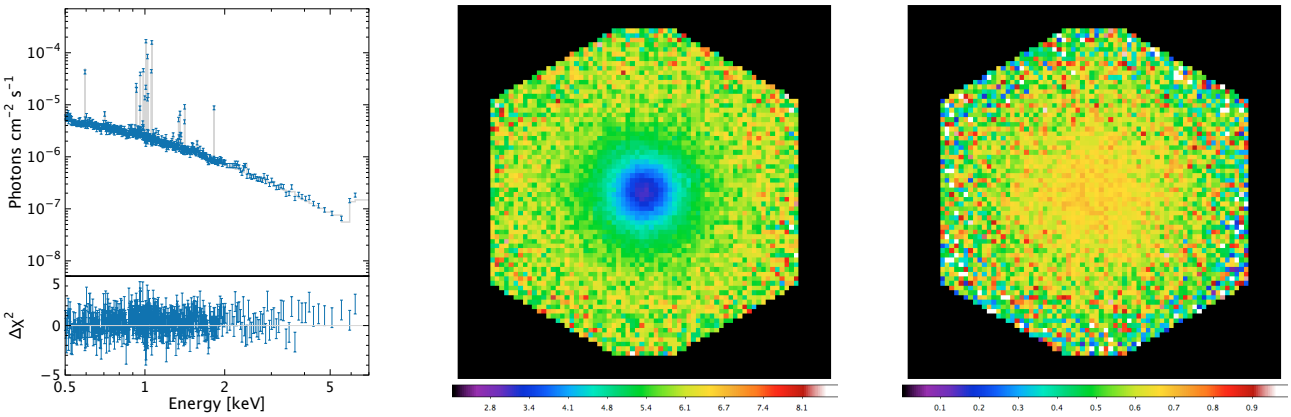

Figure 22: Results of the 3D toy model simulation with a 1 Ms exposure. **Left:** Spectrum extracted from pixel 1953 in the center of the image. The spectrum was binned so as to illustrate that for high signal to noise spectra, a single component does not properly fit the whole emission. **Center:** Reconstructed temperature map. **Right:** Reconstructed iron abundance map.

in this case, one has to make sure to include a significant portion of the cluster depth to properly account for the whole cluster emission.
We can now call `simputmulticell` to construct from this table an optimized SIMPUT catalog that will only contain a limited number of spectra taking advantage of the correlation between the different parameters in a similar way as the `simputmultispec` tool:

```
simputmulticell ParamFile=xifu_3D_grid.fits \
   ParamInputNames="T;FE_ABUND" \
   ParamNames="2;14" \
   XSPECFile=wabs_vapec_cosmo.xcm \
   InputType=TABLE \
   ParamsLogScale="no;no" \
   ParamsNumValues="100;100" \
   Elow=0.2 Eup=12.0 Estep=0.00025 \
   Emin=0.2 Emax=12.0 \
   Simput=xifu_3D.simput
```

The End-to-End simulation can then be run using the `sixtesim` tool (see §10.7.2). As can be seen from Figure 22 obtained from a 1 Ms simulation, there is significant emission mixing along the line of sight and the central region is fit with a temperature much higher than the cool core value[39]. We also see a hint from a spectrum extracted from the central pixel that for observations of much brighter clusters than this example (with higher signal-to-noise ratio), a single *vapec* component will not properly fit the spectrum. The maps presented here were obtained by blindly fitting one spectrum per pixel.

[39]The core temperature used for the simulation is not what one gets in the center when one fits a simple one-temperature model, as hot gas contributes from along the line of sight. With low signal-to-noise spectra one can describe the spectrum with a harder bremsstrahlung spectrum than what one would obtain without these contributions. This is one way how one can study the bias and systematic error in these reconstructed maps.

10.7.2 How to run and analyze an X-IFU simulation

Running the simulation

We now want to run an X-IFU simulation using either of the two SIMPUT files previously prepared. For the `clusterA2146.simput` file this can be done using the `sixtesim` tool with the following calling sequence:

```
sixtesim XMLFile=${xmldir}/xifu_nofilt_infoc.xml \
   Exposure=100000 \
   RA=239.064583333 Dec=66.3470277776 \
   EvtFile=evt_A2146.fits \
   Simput=clusterA2146.simput
```

For the `xifu_3D.simput` file you must adjust the pointing and Simput file parameters accordingly in the `sixtesim` calling sequence:

```
sixtesim XMLFile=${xmldir}/xifu_nofilt_infoc.xml \
   Exposure=1000 \
   RA=0 Dec=0 \
   EvtFile=evt_3D.fits \
   Simput=xifu_3D.simput
```

Note that we only simulate 1 ks in this example because a full 1 Ms simulation will take a few hours to complete. If the instrument files were installed in the standard directory, `$xmldir` should be set to `$SIXTE/share/sixte/instruments/athena-xifu/`. We also list below some useful optional parameters:

- `ProjCenter` Option to turn off the inside pixel position randomization during sky projection. It can sometimes be useful to simplify the mapping process. **Warning: This feature has not yet been ported to SIXTE version 3!**
- `Background` Option to turn ON and OFF the addition of non X-ray background.

Useful helper tools

Even though all SIXTE simulations output standard event files which can be processed with usual data analysis tools, we list below some helper tools available in SIXTE useful to interpret the result of X-IFU simulations:

- `makespec` Using the extended file name syntax, it is for instance possible to extract a spectrum of a specific pixel using only high resolution events by calling `makespec EvtFile="evt.fits" EventFilter="GRADING==1 && PIXID==1955"`. The `EventFilter` option makes use of the *extended filename syntax* to selected the events. The `RSPPath` can also be added to provide the path to the ARF and RMF if these files are not in the working directory.
- `imgev` Generates a count image from an Event File. In the case of the X-IFU, the typical calling sequence is (adjust `CRVAL1` and `CRVAL2` accordingly) `imgev EvtFile=evt.fits Image=image.fits CRVAL1=0.0 CRVAL2=0.0 NAXIS1=48 NAXIS2=48 CRPIX1=24.5 CRPIX2=24.5 CDELT1=-0.0015135635084518495 CDELT2=0.0015135635084518495 CoordinateSystem=0 Projection=TAN`
- `xml2svg` This tool can be used to visualize the geometry defined in a given XML file. In the case of the X-IFU one can thus see the position of the different pixels with their identification number by calling `xml2svg XMLFiles=xifu_detector_lpa25_tdm_33_275um_20211103.xml SVGName=xifu_detector.svg DrawN=3832 WriteID=yes SVGWidth=1000 Border=0.01` (see Figure 23). **Warning: This feature has not yet been ported to SIXTE version 3!**

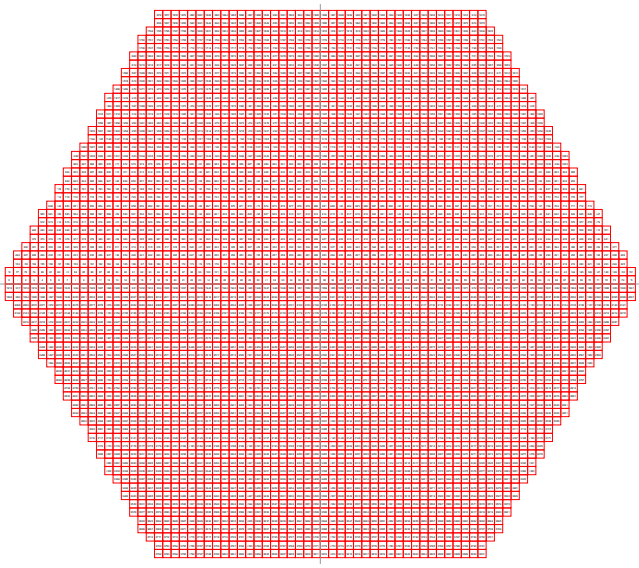

Figure 23: The baseline X-IFU geometry as created by the tool xml2svg. Each pixel is identified by a number and the grey lines show the satellite aiming point

10.7.3 How to run and analyze a simulation with tessim

As mentioned in Sect. 8.3, `tessim` is another tool implemented in SIXTE for the simulation of TES based instruments. The `tessim` tool is based on a detailed detector model and accurately simulates the physics of transition-edge sensor based microcalorimeters. This approach takes more computation time, but provides more realistic results and is thus better suited for engineering studies.
The input of `tessim` is an impact list containing photon arrival times at the sensor and their energies. The output of `tessim` is a FITS file that contains the resulting signal of the sensor at each read out time.

Creating a photon impact list with `tesgenimpacts` We can create a sample photon impact list with the SIXTE tool `tesgenimpacts`. As an example we can use the following function call to create an impact list that contains the impact of one single photon with energy 2 keV at 0.2 seconds:

```
#! /bin/bash

tesgenimpacts \
   PixImpList = piximpact_2keV.fits \
   mode = const \
   tstart = 0 \
   tstop = 0.3 \
   dtau = 200 \
   EConst = 2
```

The name of the impact list is specified by the parameter `PixImpList`. The parameter `mode` indicates that a photon is generated every `dtau`, where `dtau` is given in milliseconds. The photon generation will stop when we are over `tstop`.

Running a simulation with `tessim` Once an impact list exists we can run the simulation with the following calling sequence:

```
#! /bin/bash
```

```
tessim \
   tstart=0 \
   tstop=0.4 \
   PixType="file:LPA2.5a_tessim.fits" \
   PixImpList=piximpact_2keV.fits \
   Streamfile=tessim_output_2keV.fits \
   PixID=1 \
   triggertype = 'diff:3:100:'\
   triggersize = 9000
```

The necessary file `LPA2.5a_tessim.fits` given at the parameter `PixType` can be downloaded with

```
wget http://www.sternwarte.uni-erlangen.de/~sixte/downloads/xifu/LPA2.5a_tessim.fits
```

The file contains a large list of parameters for this type of pixel, which are necessary for simulating a TES. All these parameters can also be input on the command line, which overrides any parameter set by the given file. Here we run a simulation of 0.4 seconds. The photon hits the sensor after 200 milliseconds. The output file contains the signal of the TES split in record(s) of 9000 samples, long enough to have pulses with length equal to the baseline size of the high resolution events (8192 samples). The simulation output could also be a continuous stream of data (`triggertype=stream`) but the later reconstruction with `SIRENA` can only be performed if the data are split into records (`triggertype='diff:3:100:'`) . By default, `tessim` uses a sample rate of 156250 Hz.

Reconstruct the simulation with `SIRENA` As explained in Section 8.3.2, to reconstruct the simulation with `SIRENA` (i.e. obtain the (uncalibrated) energy of the photons), an optimal filter must be built using the noise spectrum and an average pulse template at a given energy. These steps can be achieved with these calling sequences:

1. **Noise spectrum**

   1.1 Production of noise stream impactlist

   ```
   #! /bin/bash

   tesgenimpacts \
       PixImpList = noise.piximpact \
       mode = const \
       tstart = 0 \
       tstop = 60 \
       EConst = 0. \
       dtau = 1
   ```

   1.2 Simulation of noise stream

   ```
   #! /bin/bash

   tessim \
       tstart = 0 \
   ```

```
    tstop = 60 \
    PixType = "file:LPA2.5a_tessim.fits" \
    PixImpList = noise.piximpact \
    Streamfile = tessim_noise.fits \
    PixID = 1 \
    triggertype = noise\
    triggersize = 9000
```

The simulation time must be large so that enough number of noise intervals (default is 1000) can be averaged to create the noise spectrum. The parameter `dtau` is not relevant here since no pulses are being simulated. It is also important here to select `triggertype=noise` so that the triggering of the noise stream can be performed to create different records.

1.3 Creation of the noise spectrum

```
#! /bin/bash

gennoisespec \
    inFile = tessim_noise.fits \
    outFile = tessim_noisespec.fits
```

2. **Pulse template**

2.1 Production of isolated (high resolution) pulses impact list

```
#! /bin/bash

tesgenimpacts \
    PixImpList = pulses_template_6keV.piximpact \
    mode = const \
    tstart = 0 \
    tstop = 2000 \
    EConst = 6. \
    dtau = 200
```

The simulation time in this case must be large enough so that a large number of pulses can be used for the average template. The energy of the pulses for the template and thus, for the optimal filter is 6 keV. For a more realistic simulation where the photons separations can be fixed and arrival times are not in phase with the digital samples, but with a random offset, the tool `tesconstpileup` can be used. See an example script in the SIXTE distribution under `scripts/SIRENA` folder.

2.2 Simulation of isolated pulses

```
#! /bin/bash

tessim \
    tstart = 0 \
    tstop = 2000 \
    PixType = "file:LPA2.5a_tessim.fits" \
    PixImpList = pulses_template_6keV.piximpact \
    Streamfile = tessim_pulses_template_6keV.fits \
    PixID = 1 \
```

```
    triggertype = 'diff:3:100:'\
    triggersize = 9000
```

2.3 Creation of library of optimal filter(s)

```
#! /bin/bash
tesreconstruction \
    Recordfile =  tessim_pulses_template_6keV.fits \
    TesEventFile = temp.fits \
    monoenergy = 6000. \
    LibraryFile = mylibrary_6keV.fits \
    opmode = 0 \
    NoiseFile = tessim_noisespec.fits
```

The energy of the pulses used to build the pulse template is specified by the parameter `monoenergy`, in eV and it corresponds to the energy used in the pulses simulation. The name of the library containing the optimal filter is given by the parameter `LibraryFile` and the noise spectrum previously created, through parameter `NoiseFile`. The parameter `opmode` takes value '0' for optimal filter creation and '1' for events reconstruction.

3. **Events reconstruction**

Once the optimal filter has been built, the reconstruction of the `tessim`-simulated events is done by means of the same tool used for the library creation (`tesreconstruction`), but in this case with `opmode=1`.

```
#! /bin/bash

tesreconstruction \
    Recordfile = tessim_output_2keV.fits \
    TesEventFile = recons_events_2keV.fits \
    monoenergy = 6000. \
    LibraryFile = mylibrary_6keV.fits \
    opmode = 1 \
    XMLFile=${xmlfileSX}
```

The XML file here `xmlfileSX` is required to read the different lengths (grades) with which the events will be reconstructed. Any file with the correct grading information can be used (pixel physical parameters are not relevant). For example you can use:

```
xifu_detector_lpa25_tdm_33_317um_20211029.xml
```

that can be found in directory:

```
 ${SIXTE}/share/sixte/instruments/athena-xifu/
```

Grading information can also be modified by the user editing the XML file.

The output event file will contain the full list of uncalibrated events with the format of the file produced by `sixtesim`. Inspecting this file, we can see that the energy recovered for the 2 keV photon is different from this value. Do you know why?

Exercise: Plot the current pulse contained in the output file `tessim_output_2keV.fits` of the above simulation.

Exercise: With the tessim tool `tesgenimpacts` you can also create an impact list that contains multiple photon arrival times with linearly increasing or randomly distributed energies. Have a look at the parameters of this tool by using the `plist` command:

```
plist tesgenimpacts
```

Generate an impact list that contains 20 photon arrival times separated by 50 ms with energies linearly distributed from 0.1 keV to 35 keV. Run the simulation with tessim and look at the output. Also consider the impact of noise for the different energies. Reconstruct the photon energies using the library created above with the optimal filter of 6 keV.

Exercise: Have a look at the parameters of tessim with the `plist` command. Run a simulation without noise using the impact list created in the previous exercise and look at the output. Run a simulation using the two fluid model option of the RTI transition instead of the default linear resistance model (use the pixel type 'LPA2shunt2fluid' for this simulation). How does the shape of the pulses change?

10.7.4 Used SIXTE tools in this section

The following SIXTE tools were used and explained in this section:

- `simputmultispec`
- `simputmulticell`
- `xml2svg`
- `sixtesim`
- `imgev`
- `makespec`
- `tesgenimpacts`
- `tessim`
- `gennoisespec` (SIRENA)
- `tesreconstruction` (SIRENA)

A SIMPUT Tools

A.1 Overview

The SIMPUT software package contains a set of tools enabling the generation and handling of SIMPUT files based on the SIMPUT library. The following list summarizes the available programs:

- `simputfile`
  Produces a SIMPUT file containing a catalog with a single source, a spectrum, and optionally a light curve or a PSD and an image. A sample program call is listed below. This tool constitutes a wrapper around the tools `simputsrc`, `simputspec`, `simputlc`, `simputpsd`, and `simputimg`. It can be used instead of calling the corresponding tasks individually.
- `simputsrc`
  Produces a SIMPUT file with a catalog containing a single source entry. The key properties of the source can be specified via the program parameters. The tool only produces the corresponding entry in the source catalog. It will not refer to any additional HDUs such as a spectrum. These have to be set with other tools.
- `simputspec`
  Produces a spectrum according to the specified model, stores it in the specified SIMPUT file, and links the contained source with the spectrum. The model can be either assembled by particular components or specified in an Interactive Spectral Interpretation System[40] (`ISIS`, Houck & Denicola, 2000; Houck, 2002) `*.par` file or an `XSPEC` `*.xcm` file. This tool can be used to assign a spectrum to a source produced with `simputsrc`.
- `simputlc`
  Reads an ASCII file with a light curve, stores it in the specified SIMPUT file, and links the contained source with the light curve. This tool can be used to assign a light curve to a source produced with `simputsrc`.
- `simputpsd`
  Reads an ASCII file with a PSD, stores it in the specified SIMPUT file, and links the contained source with the PSD. This tool can be used to assign a PSD to a source produced with `simputsrc`.
- `simputimg`
  Reads a FITS file with an image, stores it in the specified SIMPUT file, and links the contained source with the image. This tool can be used to assign an image to a source produced with `simputsrc`.
- `simputmerge`
  Merges multiple SIMPUT files to one file.
- `simputverify`
  Tries to open a SIMPUT file and breaks with an error message if it encounters conflicts with the file standard.
- `simputmultispec`
  Generates a SIMPUT file for an extended source with spatial variability of the flux and spectral parameters.
- `simputrotate`
  Rotate an image in a SIMPUT extension.

In order to illustrate how easy it is to generate SIMPUT files with the appropriate tools of the SIMPUT software package, we show the assembly of a file with a single point-like X-ray source located at RA = 23.8°, Dec = −12.9°. The spectral model for the source is provided in the file `model.xcm`. With these input data, the file can be generated by the following command:

```
simputfile Simput=source.simput \
   Src_ID=1 Src_Name=mysource \
```

[40] `http://space.mit.edu/cxc/isis/`

```
RA=23.8 Dec=-12.9 \
XSPECFile=model.xcm Emin=2 Emax=10
```

Of course, SIMPUT files can also be constructed by using the library routines in other programs or with an independent approach.

A.2 List of Tools

In this section we describe the parameters used by the tools in greater detail.

A.2.1 `simputfile`, `simputsrc`, `simputspec`, `simputlc`, `simputpsd`, `simputimg`

The `simputfile` task produces a SIMPUT file containing a catalog with a single source, a spectrum, and optionally a light curve or a PSD and an image. This tool constitutes a wrapper around the tools `simputsrc`, `simputspec`, `simputlc`, `simputpsd`, and `simputimg`, which set just the point source parameters, spectral shape, light curve, power spectrum, or image (for extended sources). We recommend using `simputfile` rather than these tasks for most work. The following sections describe these parameters in greater detail.
Alternatively, you can generate an initial simput file with `simputsrc` and then append spectra, images, etc., using `simputspec`, `simputlc`, etc.. These tools have the same general parameters as `simputfile`, plus a subset of the parameters listed below that pertains to each specific task. If you really want to use one of the more special tools and are unsure what parameters apply to it, use the `plist` command followed by the name of the tool to see what parameters are supported.

General parameters (`simputfile`, `simputsrc`)
`Simput`: The file name of the SIMPUT file that will be generated.
`Src_ID`: The source ID (an integer). The source ID will be written in the final event file and allows to trace each event back to the original source. This number can be arbitrary, but it is important that for a given simulation the IDs of all sources are unique. This means that if you plan to merge SIMPUT files at a later step, e.g., with `simputmerge`, you need to be careful that there is no collision between source IDs in different SIMPUT files since that tool right now does not disambiguate the names.
`Src_Name`: The name of the source (an ASCII string). Contrary to `Src_ID`, the name is not used by SIXTE, but is very useful to for a clear text identification of all sources.
`RA`, `DEC`: Right ascension and declination of the source (in degrees, J2000.0 coordinates)
`srcFlux`: The flux of the source in the energy band given by the `Emin` and `Emax` parameters, in erg $\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$.
`Emin`, `Emax`: These two parameters give the energy boundary for the `srcFlux` and other fluxes defined by parameters unless indicated otherwise.
`chatter`: An integer defining the verbosity of the program, with `0` being the lowest verbosity.
`clobber`: If set to `no`, the tool will not overwrite an already existing SIMPUT-file.
`history`: If set to `yes` or `true`, then write a history block with program parameters to the FITS header (the default).

Spectral Shape (`simputfile`, `simputspec`) The spectral shape of the source can be defined either using through simple (additive) model components consisting of a power law plus black body and a narrow and a relativistically broadened iron line, absorbed by some foreground absorption. Set the fluxes of individual components to 0 if you do not want that component to contribute to the spectral model. More complex spectra can be defined by calling ISIS or XSPEC, or by loading an ASCII file that contains a list of energies and fluxes. The spectrum in the SIMPUT file will be defined by a model grid defined by the following parameters.

Elow, Eup, Nbins: The spectrum contained in the SIMPUT file will have `Nbins` energy bins between `Elow` and `Eup`.

logEgrid: If set to `no` (the default), then the spectral bins defined by `Elow`, `Eup`, and `Nbins` will be on a linear grid, if `logEgrid` is set to `yes`, they will be on a logarithmic grid.

Note that it is required that $\texttt{Elow} \leq \texttt{Emin} < \texttt{Emax} \leq \texttt{Eup}$.

The internal spectral parameters are as follows:

plPhoIndex, plFlux: Power law photon index and flux (in the band $\texttt{Emin} \leq E \leq \texttt{Emax}$).

bbkT, bbFlux: Temperature of the black body (in keV) and its flux (again defined in the band $\texttt{Emin} \leq E \leq \texttt{Emax}$).

flFlux, flSigma: Width (σ, in keV) and total flux (in erg cm^{-2} s^{-1}) of a Gaussian line at 6.4 keV.

rflSpin, rflFlux: Parameters of a relativistic Fe Kα line at 6.4 keV, `rflSpin` is the relativistic spin parameter a ($-1 \leq a \leq 1$), `rflFlux` is the flux in the laboraory (observer) system in erg cm^{-2} s^{-1}.

NH: Hydrogen equivalent column $N_{\rm H}$ in units of H – atoms cm^{-2}.

For most sources, more complex spectral models are required. These can be generated with the spectral analysis programs ISIS or XSPEC. In order to do so, `simputfile` will invoke these tools and – if instructed to do so – first execute an arbitrary initialization code (e.g., to define local models, setup parameters needed for the model evaluation, and so on) and then load a parameter file in the format of the analysis program to setup the spectral shape. It then generates a spectrum using the spectral grid parameters defined above and writes the resulting spectrum into the SIMPUT file. Alternatively, an ASCII file containing a grid of energies and fluxes can be loaded:

ISISPrep, ISISFile: Initialization S-Lang script and spectral parameter file (`par`-file) generated by ISIS' `save_par`-command.

XSPECPrep, XSPECFile: Initialization script for XSPEC and spectral parameter file (`xcm`-file) generated by XSPEC `save model`-command.

ASCIIFile: Name of a text file containing pairs of energy (in keV) and flux (in photons s^{-1} cm^{-2} keV^{-1}) defining the spectrum. These data are interpolated onto the grid desired for the SIMPUT file.

Imaging (simputfile, simputimg) For extended sources, the flux defined in the primary SIMPUT extension is distributed using relative fluxes defined in a FITS image:

ImageFile: Name of the FITS image file

Variability: power spectra (simputfile, simputpsd) Source variability can be defined either by giving parameters for a generic power spectrum consisting of up to five Lorentzian components, an ASCII file containing a light curve (counts versus time), or an ASCII file containing a power spectrum (power versus frequency).

The internal definition of the power spectrum assumes that the source can be described by a low frequency, zero centered, Lorentzian, and then four additional Lorentzians (one designated as horizontal branch Lorentzian, and three further general Lorentzians). The shape of the Lorentzians is defined by the total rms in the Lorentzian, while the width of the component is defined by its quality factor, $Q = f/\Delta f$, where Δf is the FWHM of the component. The parameters are as follows:

LFQ, LFrms: Q of the zero-centered low frequency component.

HBOf, HBOQ, HBOrms: Frequency (Hz), Q-value, and total rms of the horizontal branch oscillation.

Q1f, Q1Q, Q1rms: Frequency (Hz), Q-value, and total rms of the first QPO Lorentzian.

Q2f, Q2Q, Q2rms: Frequency (Hz), Q-value, and total rms of the second QPO Lorentzian.

Q3f, Q3Q, Q3rms: Frequency (Hz), Q-value, and total rms of the third QPO Lorentzian.

PSDFile: Name of a text file defining the power spectrum, defined as pairs of floating point numbers designating the frequency (in Hz) and rms contribution.

Variability: light curves (`simputfile`, `simputlc`

`LCFile`: Name of a text file defining the light curve, defined as pairs of floating point numbers designating the time (in s) and relative flux (in the band defined by `Emin` and `Emax`).

`MJDREF`: Reference value for the zero point of the light curve data (Modified Julian Date).

Further parameters of `simputlc`, `simputpsd`, `simputspec`, `simputimg` The additional SIMPUT tools have the following parameters:

`Simput`: Name of the SIMPUT file to which the required extension is to be appended,

`Extname`: EXTNAME of the extension,

`Extver`: EXTVER of the extension.

A.2.2 `simputverify`

The `simputverify`-tool attempts to open and read a SIMPUT file. It will fail with a hopefully useful error message in case the file is defective.

`Simput`: Name of the SIMPUT file

`chatter`: Verbosity of the tools

A.2.3 `simputmerge`

`Simputmerge` merges two SIMPUT files. It can be used to build up source catalogues from sets of SIMPUT files for individual sources.

`Infiles`: Comma separated names of the SIMPUT files

`Outfile`: Name of the output SIMPUT file

`FetchExtensions`: If `yes` (the default), copies all extensions in the output file. If no, references to the original files are used.

`chatter`: Verbosity of the tool

`clobber`: If `no`, do not overwrite the `Outfile` if it already exists.

`history`: If `true` (the default), write a history block with program parameters to the `Outfile`.

A.2.4 `simputmultispec`

The tool `simputmultispec` generates a SIMPUT catalogue for an extended source. The spectral shape of the extended source can be described by an arbitrarily complex spectral model that is defined in an ISIS or XSPEC parameter file. The variation of certain model parameters is defined through FITS images which contain the model parameter as a function of position on the sky. The total flux at each position is defined by an additional image file. `simputmultispec` bins the parameter combinations occuring in the image onto a grid, generates spectra and images for each of these parameter combinations, and generates a SIMPUT file. See p. 71 for an example.

Not

The parameters of the tool are as follows:

`Simput`: Output simput catalog file

`srcFlux`: Total source flux (in cgs units)

`RA`, `Dec`: Right ascension and declination of the source (in degrees, J2000.0 coordinates)

`Elow`, `Eup`, `Estep`, `Nbins`, `logEgrid`: Definition of the energy grid for the spectra. Spectra are generated on a linear (`logEgrid=no`) or logarithmic (`logEgrid=yes`) grid with `Nbins` bins that extends from `Elow` to `Eup`.

ISISPrep, ISISFile: Name of the initialization script for ISIS and name of the ISIS `par`-file defining the spectral shape. Spectral parameters contained in one of the `ParamFiles` (see below) will be varied, all other values will be kept fixed at the value defined in the `par`-file.

XSPECPrep, XSPECFile: Name of the initialization script for XSPEC and name of the XSPEC `xcm`-file defining the spectral shape. Spectral parameters contained in one of the `ParamFiles` (see below) will be varied, all other values will be kept fixed at the value defined in the `xcm`-file.

ImageFile: Name of the FITS image file containing the flux distribution of the source. Note: In later simulations the fluxes in the file will be renormalized to `srcFlux`.

ParamFiles: Names of FITS image files defining the values of variable spectral parameters. The file names must be separated using semi-colons (note that when using this parameter on the command line, most shells require you to escape the semi colon with a backslash or to put the full argument into quotes).

ParamNames: semicolon separated list of parameter names or IDs corresponding to the individual `ParamFiles`.

ParamNumValues: Semicolon-separated list of the number of grid parameters for each of the parameters. Parameters will be put on a grid with `ParamNumValues` bins between the minimum and maximum value defined in the corresponding parameter image.

ParamsLogScale: list of yes and no values specifying whether the parameter grid is to be logarithmic or linear.

chatter: Verbosity of the tool

clobber: If `no`, do not overwrite the `Outfile` if it already exists.

history: If `true` (the default), write a history block with program parameters to the `Simput`.

A.2.5 `simputrotate`

`Simputrotate` rotates an image in a SIMPUT extension. The tool can be used, e.g., to add multiple rotated images of a source to generate a larger catalogue of extended sources using only a limited number of source images. The rotation is performed by moving the image coordinates assuming a central position (α_1, δ_1) to another position (α_2, δ_2). Note that these coordinates do not have to coincide with the position of the source in the SIMPUT file.

Infile: Input SIMPUT catalog file containing an image.

Outfile: Output SIMPUT catalog file which will contain the rotated image.

C1_RA, C1_Dec: right ascension, α_1, and declination, δ_1.

C2_RA, C2_Dec: right ascension, α_2, and declination, δ_2.

B SIXTE Tools

The SIXTE software package contains a set of tools to perform simulations with different X-ray instruments. The main tasks are summarized in the following list, which, however, is not comprehensive. First we are listing general tasks generic for all telescopes:

- `phogen`
  Generates a sample of X-ray photons according to the source definition in the given SIMPUT file. The process is based on routines provided by the SIMPUT library.
- `phoimg`
  Models the imaging process of X-ray photons in a Wolter-type telescope. The tool processes the list of photons produced by `phogen` and determines their impact positions on the detector.
- `gendetsim`
  Models the detection process of X-ray photons with a generic detector. The detector setup is defined in a particular XML format, as described in C. The tool processes the impact positions provided by `phoimg` and produces an event list.

- `evpat`
  Searches the event list provided by `gendetsim` for typical patterns produced by charge splitting between adjacent pixels.
- `projev`
  Determines the sky coordinates associated with the detected events.
- `pha2pi`
  Calculates the energy shift corrected PI values for PHA values in the given event file based on the specified Pha2Pi file. Each detector (and detector setting) requires its own Pha2Pi correction file. For further analysis of the PI values (e.g., makespec) the corresponding Monte Carlo RMF is required, which is provided together with the Pha2Pi file.
- `sixtesim`
  Performs a simulation using the generic instrument model. Basically the tool is a wrapper around the tasks `phogen`, `phoimg`, `gendetsim`, `evpat`, and `projev`. However, unlike these tools, it can also handle the simulation of multiple telescopes and instruments at once.
- `makespec`
  Produces a Pulse Height Amplitude (PHA) spectrum (Arnaud et al., 2009) from the event list generated by the detector simulation.
- `makelc`
  Produces a light curve (Angelini et al., 1994) from the event list generated by the detector simulation.
- `sixte_arfgen`
  Generates a corrected ARF for point sources within a user-supplied region.

The following tasks are for certain instruments (as specified elsewhere in this manual):

- `comaimg`
  Models the photon absorption and transmission process through a coded mask. The tool processes the list of photons produced by `phogen` and determines their impact positions on the detector (Oertel, 2013).
- `comadet`
  Operates a specialized detector model for the HXI detector of MIRAX. The tool processes the impact positions provided by `comaimg` and produces an event list (Oertel, 2013).
- `comarecon`
  Reconstructs the positions of the illuminating sources based on the shadow pattern on the detector, which is arising from the absorption by the coded mask. The tool processes the event list provided by `comadet` (Oertel, 2013).

The following example of a simulation with a model of one of the seven eROSITA sub-instruments illustrates the usage of these tools:

```
sixtesim \
   XMLFile=erosita.xml \
   Attitude=allskysurvey.att \
   Exposure=86400.0 \
   Simput=source.simput \
   RawData=raw.fits \
   EvtList=events.fits
```

This program call initiates a simulation of a 1-day interval of the eROSITA all-sky survey based on the attitude specified in the FITS file `allskysurvey.att`. The model of the instrument is defined in the XML file `erosita.xml`. The observed X-ray source is described in the SIMPUT file `source.simput`. The output of the simulation run are two event lists: One is stored in the FITS file `raw.fits` and contains the raw single-pixel events. The other one is stored in the FITS file `events.fits` and contains the recombined patterns of split events.

In order to simultaneously simulate all seven sub-instruments of eROSITA, multipel `XMLFiles` need to be supplied

C XML Instrument Configuration

One of the key features of SIXTE is the possibility to enable simulations for a wide range of different X-ray instruments. The therefore necessary flexibility is achieved by using a generic telescope and detector module, which can be configured using a specific XML format. This generic approach has proven invaluable both for developing a wide coverage of various existing instruments with different operation modes and for implementations of new instrumentation in the course of several assessment studies. Especially for the latter purpose the XML format is very convenient for studying modifications of an instrument setup.
With an appropriate combination of these commands, most kinds of current X-ray instruments can be described, as shown with the examples above. For specific instrument models such as the LAD aboard *LOFT*, particular XML tags are used, which do not apply to the generic module and are therefore not part of this list.
In Sect. 6 we illustrate the power of this concept giving instrument specific details for the setup of eROSITA, *Athena*, *XMM-Newton*, or, *Suzaku*. In the following we give a list of the available XML tags and their meaning.
An entire instrument must be encapsulated in the tag
`<instrument telescop="..." instrume="..."> ... </instrument>`
With the `telescop` and `instrume` values being identifier strings that are also written into all SIXTE output files.

C.1 Telescope

The definition of the telescope is encapsulated by the tag
`<telescope number="...">...</telescope>`
The `number` attribute here is optional when simulating with only one XML file but mandatory when simulating multiple telescopes at once, such as `eROSITA`, where it is used to enumerate the various output files.
The following telescope-specific tags should be contained within this environment.

- `<fov diameter="..."/>`
  Diameter of the FOV of the instrument in [deg].
- `<focallength value="..."/>`
  Focal length of the telescope in [m].
- `<vignetting filename="..."/>`
  File containing the vignetting function of the telescope (George & Zellar, 1994).
- `<psf filename="..."/>`
  File containing the 2-dimensional PSF of the telescope (George & Yusaf, 1995).
- `<arf filename="..."/>`
  File containing the instrument ARF (George et al., 1998, 2007), which includes the effective area of the telescope, the quantum efficiency of the detector, the transmission of possible filters, etc.

C.2 Detector

Information specific to detectors is encapsulated in the tag
`<detector type="..." chip="..."> ..</detector>`
The `type` attribute refers to the type of the detector, which may itself require specific information, as will be outlined below.

The `chip` attribute is optional, and can be used to identify a specific detector. This is especially relevant when specifiying an instrument with one telescope and multiple detectors, such as the *Athena WFI*.

C.2.1 General Tags

The following tags apply to all detectors, regardless of type.

- `<rmf filename="..."/>`
  File containing the detector redistribution matrix RMF (George et al., 1998, 2007), which represents the energy resolution.
- `<threshold_readout_lo_keV value="..."/>`
  Lower signal readout threshold in [keV].
- `<threshold_readout_up_keV value="..."/>`
  Upper signal readout threshold in [keV].
- `<threshold_event_lo_keV value="..."/>`
  Lower threshold for an event detection in [keV].
- `<phabackground filename="..." lightcurve="..." vignetting="..."/>`
  PHA file containing the background spectrum. During the simulation random background events are produced with energies according to the distribution given in the PHA file. Additionally, a light curve and vignetting file which should be considered for the background events can be specified with the optional attributes `lightcurve` and `vignetting`.
- `<auxbackground filename="..." lightcurve="..."/>`
  File containing a list of individual background events, which originate, e.g., from cosmic-ray photons. During the simulation randomly selected entries from this list are inserted as background. Additionally, a light curve can be specified with the optional attribute `lightcurve`.
- `<pha2pi filename="..."/>`
  File containing the energy shift corrected PI values for each PHA channel. If specified, the pha2pi correction is automatically performed when simulating events, i.e., resulting event files will contain corrected PI values.

Furthermore, there are tags allowing the creation of loops, meaning the repetition of XML tags in a structured form:

- `<loop start="..." end="..." increment="..." variable="...">...</loop>`
  Repeat the contained XML tags, while the loop variable is increased from its starting to its end value by the specified increment.
- `<hexagonloop radius="..." pixelpitch="..." cross="..."> ... </hexagonloop>`
  Repeat the contained XML tags while paving an hexagon having the same area as a circle of the given `radius` with a pitch in both directions given by `pixelpitch`. `cross` should be 0 to use a grid containing the origin point and 1 otherwise. Inside the loop, the variables `$p` can be used to access the `pixelpitch` value whereas `$x` and `$y` will contain the current x and y position in the loop.

These tags may be useful to define the focal plane geometry of an instrument or its readout.

C.2.2 Geometry Tags

Currently, SIXTE supports two type of focal plane geometries, specified via the tag
`<geometry type="..."> ... </geometry>`

- The `rectarray` geometry, used to specify pixels on a rectangular grid. This is used for, e.g., Silicon based detectors such as a CCD or DEPFET
- The `free` geometry, made up of rectangular pixels with arbitrary placement. Used for, e.g., the *Athena X-IFU*.

If no geometry is specified, SIXTE will default to the `rectarray` geometry.
Both geometries may use the WCS tag:
`<wcs xrpix="..." yrpix="..." xrval="..." yrval="..." xdelt="..." ydelt="..." rota="..."/>`
which defines the focal plane coordinate system analogous to the WCS keywords introduced by Greisen & Calabretta (2002) and Calabretta & Greisen (2002). The entries `xrpix` and `yrpix` define the reference pixel, the entries `xrval` and `yrval` the corresponding location in the focal plane, and `xdelt` and `ydelt` the dimensions of the pixels. Units are [m]. Additionally, the `rota` key, with values given in degrees, may be used to rotate the focal plane geometry around the point given by `xrval` and `yrval`. Refer to Fig. 10 for an illustration.
In the `rectarray` geometry, pixels are then placed along the grid defined by the WCS tag. For the `free` geometry, only the keys `xrval`, `yrval` and `rota` are used, as no pixel grid is implied.

The `rectarray` geometry requires two additional tags

- `<dimensions xwidth="..." ywidth="..."/>`
  Number of pixels in x- and y-direction.
- `<pixelborder x="..." y="..."/>`
  Border width of the pixels in [m].

The `free` geometry also requires more information in its containing tag:
`<geometry type="free" npix="..." xoff="..." yoff="..."> ... </geometry>`

- `xoff` and `yoff` specify an additional offset from the focal point.
- `npix` specifies the number of pixels in the array.

Within the `free` geometry, pixels can then be placed via the `pixel` tag:

```
<pixel>
  <shape posx="..." delx="..." posy="..." dely="..." width="..." height="..."/>
</pixel>
```

The central position of a pixel in meters is then calculated as

$$\texttt{x} = \texttt{posx} * \texttt{delx}$$
$$\texttt{y} = \texttt{posy} * \texttt{dely}$$

The length of the pixel in X direction is the `width` and in Y direction the `height`, both given in meters.
After all pixels have been specified, overlapping pixels will be removed, with pixels placed later taking priority. The number of resulting pixels is then checked against the `npix` value.
We note that because this focal plane description can incorporate gaps between the pixels, it is necessary to use in the generic instrument definition XML an ARF that does not contain the filling factor contribution.

C.2.3 Type-Specific Tags

Silicon-based detectors of type `ccd` or `depfet` require similar tags:

- `<cte value="..."/>`
  CTE applied at a single line shift in a CCD device. The value must lie in the range from 0 to 1.
- `<split type="..." par1="..." par2="..."/>`
  Model for the charge cloud splitting between adjacent pixels. Currently available are a Gaussian (`GAUSS`) and an exponential model (`EXPONENTIAL`) developed for eROSITA by K. Dennerl (priv. comm.).

- `<threshold_split_lo_fraction value="..."/>`
  Lower threshold for a split partner as a fraction of the total signal of the pattern.
- `<threshold_split_lo_keV value="..."/>`
  Lower threshold for charge cloud contributions to the event recombination. Split partners less this value will be ignored.
- `<readout mode="...">...</readout>`
  Detector operation mode. A time-triggered (`TIME`) and an event-triggered (`EVENT`) mode are available. The event-triggered mode requires additional information in the form of a `samplefreq` attribute giving the sampling frequency of the detector. For the time-triggered mode the definition of the readout sequence is defined by the encapsulated XML tags (usually within a `loop` tag).
  - `<wait time="..."/>`
    Define a period of inactivity in [s]. During that time the detector is exposed to incident radiation.
  - `<lineshift/>`
    Used for CCDs. Shift the collected charges by one line towards the readout anodes.
  - `<readoutline lineindex="..." readoutindex="..."/>`
    Read out a particular line of the pixel array and assign the specified line number, which might be different from the actual line index (e.g. for shifted lines in a CCD).
  - `<clearline lineindex="..."/>`
    Clear the signals in a particular line of the pixel array without recording them.
  - `<newframe/>`
    Define the beginning of a new readout frame.

A `depfet` detector further requires its own tag
`<depfet integration="..." clear="..." settling_1="..." settling_2="..." type="..."/>`
, whose values are explained in Sect. 7.2

Microcalorimeters of type `microcal` also use the above `readout` tag, only allowing the `EVENT` mode. Further information with regards to their event reconstruction are within the
`<reconstruction>...</reconstruction>`
tag, specifically

- `<grading num="..." name="..." pre="..." post="..." rmf="..."/>`
  The grading information, usually for multiple grades, as explained in Sect. 8.1.1
- `<crosstalk> ... </crosstalk>`, containing information on the Crosstalk
  - `<mux type="..." channel_freq_list="..."/>`
    Information on the multiplexing, currently only supporting the type `tdm` (Time-Division Multiplexing), with a `channel_freq_list`, containing a filename with channel information for each pixel.
  - `<propcrosstalk filename="..." scaling1="..." scaling2="..." scaling3="..." trig_thresh="..."/>`
    Proportional crosstalk, with a `filename` pointing to a lookup table file, three scaling parameters (the following pixel, its following pixel and the entire row), and a `trig_thresh` value, where a crosstalk event can create a "fake" trigger if the scaling multiplied by the perpetrator energy exceeds this value.
  - `<dercrosstalk filename="..." scaling="..." trig_thresh="..."/>`
    Derivative crosstalk, with a `filename` pointing to a lookup table file, one scaling parameter (the following pixel is affected), and a `trig_thresh` value, same as for proportional crosstalk
  - `thermalcrosstalk`
    Thermal crosstalk, containing multiple tags of the format

`<pair distance="..." weight="..." timedepfile="..." trig_thresh="..."/>`
Each `pair` tag indicates the crosstalk for a given pixel-to-pixel `distance` in meters, with a given timedependency table in the `timedepfile` attribute and scaling `weight`. `trig_thresh` functions as in the other crosstalk types.

D Charge Cloud Models

As indicated in Sect. 6, SIXTE provides two models to simulate the distribution of charge between neighboring pixels in a silicon-based pixelized detector (Schmid, 2012).

D.1 Gaussian Charge Clouds

The Gaussian model assumes a 2-dimensional rotationally symmetric charge cloud. The charge fraction collected in the detector pixel with the corners (x_n, y_m), (x_{n+1}, y_m), (x_{n+1}, y_{m+1}), and (x_n, y_{m+1}) can be determined as (Popp, 2000):

$$c_{n,m} = \frac{c_{\mathrm{total}}}{2\pi\sigma^2} \cdot \int_{x_n}^{x_{n+1}} \mathrm{e}^{-\frac{(x-x_\mathrm{i})^2}{2\sigma^2}}\,\mathrm{d}x \int_{y_m}^{y_{m+1}} \mathrm{e}^{-\frac{(y-y_\mathrm{i})^2}{2\sigma^2}}\,\mathrm{d}y \tag{14}$$

where $(x_\mathrm{i}, y_\mathrm{i})$ denotes the impact position of the photon and c_{total} the total generated charge. As the charge cloud size ($\sigma \sim 10\,\mu$m) is typically smaller than the size of the detector pixels, Eq. (14) is evaluated only for the four pixels directly surrounding the impact position. The numerical calculation is performed with the function `gsl_sf_erf_Q()` of the GNU Scientific Library (GSL)[41]. By adjusting the size of the charge cloud and the lower detection threshold for split partners, the simulated pattern distribution can be adjusted to fit measurements with real detectors, as shown in Fig. 24 for the EPIC pn camera on *XMM-Newton*. The charge cloud size $\sigma =$ `par1` $+ \sqrt{E} \cdot$ `par2` in the GAUSS model can be energy-dependent. `par1` is the charge cloud size (in units of meters) at a photon energy of 1 keV. `par2` represents the slope of the function giving the charge cloud size versus photon energy.

D.2 Charge Split Model

In the charge split model developed by K. Dennerl (priv. comm.) for eROSITA the relative charge distribution among the four pixels surrounding the impact position (indices as above) is obtained as:

$$c^\star_{n,m} = \exp\left[-\left(\frac{r_{n,m}}{0.355}\right)^2\right] \tag{15}$$

with

$$r_{n,m} = \sqrt{\left(x_\mathrm{i} - x_n - \frac{d_x}{2}\right)^2 + \left(y_\mathrm{i} - y_m - \frac{d_y}{2}\right)^2} \tag{16}$$

After normalization, the values determined by Eq. (15) constitute the relative distribution of the generated charge among the four neighboring pixels.

$$c_{n,m} = \frac{c^\star_{n,m}}{\sum_i \sum_j c^\star_{i,j}} \tag{17}$$

With this model the observed pattern distribution can be reproduced.

[41] `http://www.gnu.org/software/gsl/`

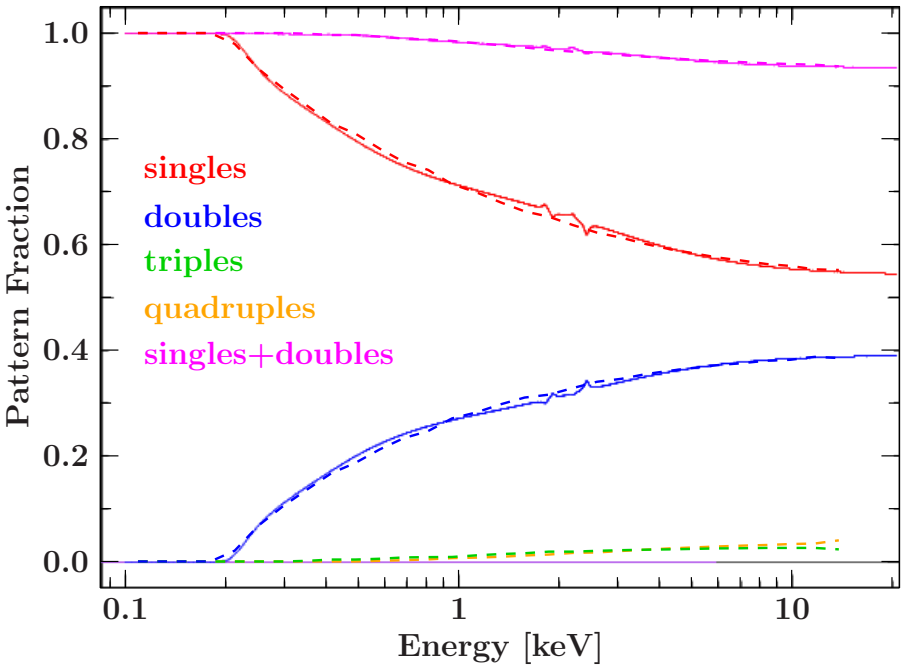

Figure 24: Comparison of the Gaussian charge cloud model (dashed lines) with the distribution of pattern types obtained with the SAS task `epatplot` (straight lines) for the EPIC pn camera on *XMM-Newton* operated in Small Window mode. The simulated distribution matches very well the values expected from the instrument model.

E Crosstalk Implementation for Calorimeter Detectors

As has been mentioned in Sect. 8, the simulation for calorimeter detectors includes the possibility to predict cross talk events expected for a chosen configuration. We currently distinguish between three different types of crosstalk, namely *thermal crosstalk*, *electrical crosstalk*, and *intermodulation crosstalk*. A detailed description of each type can be found below in the respective sections. In the following we will give a brief overview how generally crosstalk events are treated in SIXTE.
All crosstalk mechanisms are treated seperately. However, once the existence and strength of a crosstalk event is known, the remaining procedure is the same.
For each pixel in the sensor, SIXTE stores a list of *victim pixels*, meaning pixels that can receive crosstalk from a given *perpetrator pixel*. When an event is registered in a pixel, each of its victims receive a so-called crosstalk proxy, notifiying that a possible perpetrator has received an event of a given energy and at a given time.
When an event is read out, SIXTE collects all the crosstalk proxies that could affect this event and calculates their influence on the measured energy, using lookup table for the given crosstalk mechanism which depends on the energies of the victim and perpetrator event as well as their difference in time.
There is one relatively rare interaction that can happen aside from a simple energy shift: Depending on the crosstalk mechanism, the crosstalk signal in a victim pixel by itself can be strong enough to lead to the triggering of a fake event. The threshold for this case is saved per crosstalk mechanism as the `trig_thresh` parameter in the XML. These fake events can not cause crosstalk by themselves, however.

E.1 Thermal Crosstalk

Thermal crosstalk may occur due to the thermal coupling between two pixels. As such, its magnitude is based on the distance between two pixels, rapidly falling of with distance.
In the SIXTE XMLs, a separate lookup table is saved for a set of distances, usually for three pairs:

- Direct neighbours (a distance of one pixel width)
- Diagonal neighbours (a distance of *sqrt*2 pixel widths)
- Second order neighbours (a distance of two pixel widths)

E.2 Electrical Crosstalk

In the current implementation, electrical crosstalk can happen via two mechanisms:

- As a result of the multiplexing scheme (Time-Division Multiplexing)
- Due to inductive coupling between pixels

The former effect causes a crosstalk signal proportional in shape to the perpetrator signal affecting the next pixel to be read out, the one after it and the entire channel (by different magnitudes). The latter causes a crosstalk signal proportional to the derivative of the perpetrator signal in two "wiring neigbhors", which are the pixels read out before and after the perpetrator.
Aside from the affected pixels, the crosstalk types are handled the same way internally by interpolating from a lookup table.

List of Acronyms

APE: All Purpose Parameter Environment
ARF: Ancillary Response File
Athena: Advanced Telescope for High ENergy Astrophysics
CCD: Charge Coupled Device
CTE: Charge Transfer Efficiency
DEPFET: DEPleted Field Effect Transistor
ECAP: Erlangen Centre for Astroparticle Physics
EPIC: European Photon Imaging Camera
eROSITA: extended Röntgen Survey with an Imaging Telescope Array
FITS: Flexible Image Transport System
FOV: Field of View
FWHM: Full Width at Half Maximum
GUI: Graphical User Interface
GSFC: Goddard Space Flight Center
GSL: GNU Scientific Library
GTI: Good Time Interval
HDU: Header and Data Unit
HEASARC: High Energy Astrophysics Science Archive Research Center
HEASOFT: High Energy Astronomy software
HXI: Hard X-ray Imager
ISIS: Interactive Spectral Interpretation System
IXO: International X-ray Observatory
LAD: Large Area Detector
LOFT: Large Observatory For x-ray Timing
NuSTAR: Nuclear Spectroscopic Telescope ARray
MIRAX: Monitor e Imageador de RAios-X
NASA: National Aeronautics and Space Administration
OOT: Out Of Time
PHA: Pulse Height Amplitude
PIL: Parameter Interface Library
PSD: Power Spectral Density
PSF: Point Spread Function
QPO: Quasi-Periodic Oscillation
RMF: Redistribution Matrix File
SASS: Standard Analysis Software System
SIRENA: Software Ifca to Reconstruct photon ENergies for Athena
SIXTE: SImulation of X-ray TElescopes
SIMPUT: SIMulation inPUT
SRG: Spectrum-Roentgen-Gamma
TES: Transition Edge Sensor
WCS: World Coordinate System
WFI: Wide Field Imager
WFM: Wide Field Monitor
X-IFU: X-ray Integral Field Unit
XIS: X-Ray Imaging Spectrometer

XML: eXtensible Markup Language
XMM-Newton: X-ray Multi-Mirror Mission Newton

References

Angelini L., Pence W., Tennant A.F., 1994, The Proposed Timing FITS File Format for High Energy Astrophysics Data, OGIP/93-003, HEASARC, Greenbelt, USA

Arnaud K.A., 1996, In: G. H. Jacoby & J. Barnes (ed.) Astronomical Data Analysis Software and Systems V, Vol. 101. Astronomical Society of the Pacific Conference Series, p. 17

Arnaud K.A., George I.M., Tennant A.F., 2009, The OGIP Spectral File Format, OGIP/92-007, HEASARC, Greenbelt, USA

Barcons X., Bregman J., Ohashi T., et al., 2011a, IXO Assessment Study Report, Technical report, ESA, Paris

Barcons X., Lumb D.H., Nandra K., et al., 2011b, Athena Assessment Study Report, Technical report, ESA, Paris

Barret D., den Herder J.W., Piro L., et al., 2013, arXiv:1308.6784

Bautz M.W., Kissel S.E., Prigozhin G.Y., et al., 2004, In: Holland A.D. (ed.) High-Energy Detectors in Astronomy, Vol. 5501. Proc. of SPIE, p.111

Bookbinder J., 2010, In: Arnaud M., Murray S.S., Takahashi T. (eds.) Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray, Vol. 7732. Proc. of SPIE, p. 1B

Bookbinder J., Barret D., Bautz M., et al., 2010, The International X-ray Observatory - RFI, Technical report, Smithsonian Astrophysical Observatory, Cambridge

Borkowski J., Lock T., Walter R., 2002, Parameter Interface Library Users Manual (v. 1.8.5), INTEGRAL Science Data Centre, Versoix, Switzerland

Braga J., Mejía J., 2006, In: Turner M.J.L., Hasinger G. (eds.) Space Telescopes and Instrumentation II: Ultraviolet to Gamma Ray, Vol. 6266. Proc. of SPIE, p. 0M

Brandt S., Hernanz M., Alvarez L., et al., 2012, In: Takahashi T., Murray S.S., den Herder J.W. (eds.) Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray, Vol. 8443. Proc. of SPIE, p. 2G

Calabretta M.R., Greisen E.W., 2002, Astronomy and Astrophysics 395, 1077

Dauser T., Falkner S., Lorenz M., et al., 2019, Astronomy and Astrophysics 630, A66

Deák I., 1990, Random Number Generators and Simulation, Mathematical Methods of Operations Research 4, Akadémiai Kiadó, Budapest

Feroci M., Stella L., Vacchi A., et al., 2010, In: Arnaud M., Murray S.S., Takahashi T. (eds.) Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray, Vol. 7732. Proc. of SPIE, p. 1V

Feroci M., Stella L., van der Klis M., et al., 2012, Experimental Astronomy34, 415

Finoguenov A., Tanaka M., Cooper M., et al., 2015, Astronomy and Astrophysics 576, A130

Gabriel C., Ibarra Ibaibarriaga A., Hoar J., 2005, In: Siegmund O.H.W. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XIV, Vol. 5898. Proc. of SPIE, p.469

George I.M., Angelini L., 1995, Specification of Physical Units within OGIP FITS files, OGIP/93-001, NASA/GSFC, Greenbelt, USA

George I.M., Arnaud K.A., Ruamsuwan B.P.L., Corcoran M.F., 1998, The Calibration Requirements for Spectral Analysis, HEASARC, Greenbelt, USA

George I.M., Arnaud K.A., Ruamsuwan B.P.L., Corcoran M.F., 2007, The Calibration Requirements for Spectral Analysis, HEASARC, Greenbelt, USA

George I.M., Yusaf R., 1995, The OGIP Format for 2-D (Image) Point Spread Function Datasets, CAL/GEN/92-027, HEASARC, Greenbelt, USA

George I.M., Zellar R., 1994, The OGIP Format for 'Vignetting' Functions, CAL/GEN/92-021, HEASARC, Greenbelt, USA

Gould H., Tobochnik J., Christian W., 2006, An introduction to computer simulation methods: Applications to Physical Systems (third edition), Addison-Wesley, San Francisco

Greisen E.W., Calabretta M.R., 2002, Astronomy and Astrophysics 395, 1061

Grindlay J., Allen B., Angelini L., et al., 2011, The Hard X-ray Imager for MIRAX, Technical report, NASA, Washington

Hamaguchi K., Corcoran M.F., Russell C.M.P., et al., 2014, Astrophysical Journal 784, 125

Hanisch R.J., Farris A., Greisen E.W., et al., 2001, Astronomy and Astrophysics 376, 359

Harrison F.A., Boggs S., Christensen F., et al., 2010, In: Arnaud M., Murray S.S., Takahashi T. (eds.) Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray, Vol. 7732. Proc. of SPIE, p. 0S

Houck J.C., 2002, In: G. Branduardi-Raymont (ed.) High Resolution X-ray Spectroscopy with XMM-Newton and Chandra.

Houck J.C., Denicola L.A., 2000, In: N. Manset, C. Veillet, & D. Crabtree (ed.) Astronomical Data Analysis Software and Systems IX, Vol. 216. Astronomical Society of the Pacific Conference Series, p.591

Irwin K.D., Hilton G.C., 2005, Transition-Edge Sensors. In: Enss C. (ed.) Cryogenic Particle Detection. Springer, Berlin, Heidelberg, p. 63

Jansen F., Lumb D., Altieri B., et al., 2001, Astronomy and Astrophysics 365, L1

Kendziorra E., Bihler E., Colli M., et al., 1998, In: Siegmund, O. H. W. and Gummin, M. A. (ed.) EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy IX, Vol. 3445. Proc. of SPIE, p.50

Kendziorra E., Bihler E., Grubmiller W., et al., 1997, In: Siegmund, O. H. and Gummin, M. A. (ed.) EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy VIII, Vol. 3114. Proc. of SPIE, p.155

Kendziorra E., Colli M., Kuster M., et al., 1999, In: Siegmund, O. H. W. and Flanagan, K. A. (ed.) EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy X, Vol. 3765. Proc. of SPIE, p.204

Koyama K., Tsunemi H., Dotani T., et al., 2007, Publications of the Astronomical Society of Japan 59, 23

Kuster M., Benlloch S., Kendziorra E., Briel U.G., 1999, In: Siegmund, O. H. and Flanagan, K. A. (ed.) EUV, X-Ray, and Gamma-Ray Instrumentation for Astronomy X, Vol. 3765. Proc. of SPIE, p.673

Lehmer B.D., Brandt W.N., Alexander D.M., et al., 2005, apjs 161, 21

Madsen K., Alexander D., Bhalerao V., et al., 2011, In: HEAD meeting no. 12, Vol. 12. AAS, p. 43.08

Martin M., 2009, Dissertation, Eberhard-Karls-Universität Tübingen, Tübingen

Meidinger N., Andritschke R., Ebermayer S., et al., 2009, In: UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XVI, Vol. 7435. Proc. of SPIE, p. 02

Meidinger N., Andritschke R., Ebermayer S., et al., 2010, Nuclear Instruments and Methods in Physics Research A 624, 321

Meidinger N., Andritschke R., Elbs J., et al., 2011, In: Tsakalakos L. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XVII, Vol. 8145. Proc. of SPIE, p. 02

Meidinger N., Andritschke R., Elbs J., et al., 2008, In: Turner M.J.L., Flanagan K.A. (eds.) Space Telescopes and Instrumentation 2008: Ultraviolet to Gamma Ray, Vol. 7011. Proc. of SPIE, p. 0J

Meidinger N., Andritschke R., Hälker O., et al., 2006, In: High Energy, Optical, and Infrared Detectors for Astronomy II, Vol. 6276. Proc. of SPIE, p. 18

Meidinger N., Andritschke R., Hälker O., et al., 2007, In: Siegmund O.H.W. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XV, Vol. 6686. Proc. of SPIE, p. 0H

Nandra K., Barret D., Barcons X., et al., 2013, arXiv:1306.2307

Oertel M., 2013, Diplomarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg

Pavlinsky M., Hasinger G., Parmar A., et al., 2006, In: Turner M.J.L., Hasinger G. (eds.) Space Telescopes and Instrumentation II: Ultraviolet to Gamma Ray, Vol. 6266. Proc. of SPIE, p. 0O

Pavlinsky M., Sunyaev R., Churazov E., et al., 2008, In: Turner M.J.L., Flanagan K.A. (eds.) Space Telescopes and Instrumentation 2008: Ultraviolet to Gamma Ray, Vol. 7011. Proc. of SPIE, p. 0H

Pavlinsky M., Sunyaev R., Churazov E., et al., 2009, In: O'Dell S.L., Pareschi G. (eds.) Optics for EUV, X-Ray,

and Gamma-Ray Astronomy IV, Vol. 7437. Proc. of SPIE, p. 08
Pence W., 1999, In: D. M. Mehringer, R. L. Plante, & D. A. Roberts (ed.) Astronomical Data Analysis Software and Systems VIII, Vol. 172. Astronomical Society of the Pacific Conference Series, p. 487
Pence W.D., Chiappetti L., Page C.G., et al., 2010, Astronomy and Astrophysics 524, A42
Ponz J.D., Thompson R.W., Munoz J.R., 1994, Astronomy and Astrophysics Supplement Series 105, 53
Popp M., 2000, Ph.D. thesis, Ludwig-Maximilians-Universität München
Predehl P., 2012, In: Takahashi T., Murray S.S., den Herder J.W. (eds.) Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray, Vol. 8443. Proc. of SPIE, p. 1R
Predehl P., Andritschke R., Becker W., et al., 2011, In: Tsakalakos L. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XVII, Vol. 8145. Proc. of SPIE, p. 0D
Predehl P., Andritschke R., Böhringer H., et al., 2010a, In: Arnaud M., Murray S.S., Takahashi T. (eds.) Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray, Vol. 7732. Proc. of SPIE, p. 0U
Predehl P., Andritschke R., Bornemann W., et al., 2007, In: Siegmund O.H.W. (ed.) UV, X-Ray, and Gamma-Ray Space Instrumentation for Astronomy XV, Vol. 6686. Proc. of SPIE, p. 17
Predehl P., Böhringer H., Brunner H., et al., 2010b, AIP Conf. Proc. 1248, 543
Predehl P., Hasinger G., Böhringer H., et al., 2006, In: Turner M.J.L., Hasinger G. (eds.) Space Telescopes and Instrumentation II: Ultraviolet to Gamma Ray, Vol. 6266. Proc. of SPIE, p. 0P
Rau A., Meidinger N., Nandra K., et al., 2013, arXiv:1308.6785
Russell H.R., McNamara B.R., Sanders J.S., et al., 2012, Monthly Notices of the Royal Astronomical Society 423, 236
Schmid C., 2012, Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg
Schmid C., Brand, T. Kuehnel M., Wille M., et al., 2012, In: Goldwurm A., Lebrun F., Winkler C. (eds.) An INTEGRAL view of the high-energy sky (the first 10 years) – 9th INTEGRAL Workshop and celebration of the 10th anniversary of the launch, Vol. INTEGRAL 2012. Proc. of Science, p. 152
Schmid C., Martin M., Wilms J., et al., 2010, AIP Conf. Proc. 1248, 591
Schmid C., Smith R., Wilms J., 2013, SIMPUT – A File Format for Simulation Input, Technical report, HEASARC, Cambridge
Schmid C., Wilms J., Oosterbroek T., et al., 2011, In: Barret D., Mendez M., Paltani S. (eds.) Fast X-ray timing and spectroscopy at extreme count rates, Vol. HTRS 2011. Proc. of Science, p. 070
Seward F.D., Butt Y.M., Karovska M., et al., 2001, Astrophysical Journal 553, 832
Strüder L., Briel U., Dennerl K., et al., 2001, Astronomy and Astrophysics 365, L18
Szymkowiak A., Kelley R., Moseley S., Stahle C., 1993, J. Low Temp. Phys. 93, 281
Timmer J., König M., 1995, Astronomy and Astrophysics 300, 707
Vikhlinin A., Kravtsov A., Forman W., et al., 2006, Astrophysical Journal 640, 691
Weisskopf M.C., Brinkman B., Canizares C., et al., 2002, Publications of the Astronomical Society of the Pacific 114, 1
Wells D.C., Greisen E.W., Harten R.H., 1981, Astronomy and Astrophysics Supplement Series 44, 363
Wise M.W., Huenemoerder D.P., Davis J.E., 1997, In: Hunt G., Payne H. (eds.) Astronomical Data Analysis Software and Systems VI, Vol. 125. Astronomical Society of the Pacific Conference Series, p. 477
Zane S., Walton D., Kennedy T., et al., 2012, In: Takahashi T., Murray S.S., den Herder J.W. (eds.) Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray, Vol. 8443. Proc. of SPIE, p. 2F
Zoglauer A., Kruse-Madsen K., Kitaguchi T., et al., 2011, In: HEAD meeting no. 12, Vol. 12. AAS, p. 43.07