Abstract

This document contains a tutorial on three example simulations for XRISM with SIXTE. First, in Sect. 2.1, we show how to create a simple point source SIMPUT file and then use this to run a basic simulation with Xtend and Resolve. Section 2.2 goes more into detail with bright point-like sources and how to analyze them in Sixte. Finally, in Sect. 2.3 we simulate the first light observation of N132D.
Besides that, you can find a detailed tutorial with in-depth explanations of all features in the SIXTE manual `https://www.sternwarte.uni-erlangen.de/~sixte/data/simulator_manual.pdf`. If you have questions or in case you encounter problems, you can always contact `sixte-support@lists.fau.de` and we will try our best to help you.

Change Record

Issue	*Date*	*Description of Change*	*Affected Pages*
1	2024 Feb 12	XRISM Workshop Geneva (Initial Release)	All

Approvals

Function	*Name*	*Date*	*Signature*
Author	L. Dauner	N/A	N/A
Author	T. Dauser	N/A	N/A
Author	C. Kirsch	N/A	N/A
Author	M. Lorenz	N/A	N/A
Author	J. Wilms	N/A	N/A

Contents

1 Installation

1.1 Software

You can install and run SIXTE locally on Linux and MacOS, or use the software through the JHU SciServer environment (via the `sixte_users` group). For either case, see `https://www.sternwarte.uni-erlangen.de/sixte/installation/`

1.2 Instrument Files

The XRISM instrument files for SIXTE are available at `https://www.sternwarte.uni-erlangen.de/~sixte/downloads/xrism_workshop/instruments_xrism-0.0.2.tar.gz`. Move the files to your SIXTE installation directory and unpack them in the existing data tree. You can do this with the following commands:

```
mv instruments_xrism-0.0.2.tar.gz $SIXTE/
cd $SIXTE
tar -zxvf instruments_xrism-0.0.2.tar.gz
```

2 Example Simulations

All necessary files for the simulations can be downloaded here: `https://www.sternwarte.uni-erlangen.de/~sixte/downloads/xrism_workshop/`. We note that if you are using the SciServer instead of a local installation (see Sect. 1), those files are already uploaded there at

```
$HOME/workspace/Storage/sixte/sixte_volume/sixtedata/2024_XRISM_Workshop_Geneva
```

You just have to initialize SIXTE on the SciServer with

```
source $HOME/workspace/Storage/sixte/sixte_volume/sixte_setup.sh
```

and then you can directly start simulating.

2.1 Point Source

2.1.1 Step 1: Create SIMPUT

First, we will simulate an observation of a point source with Xtend and Resolve. To create the simulation input SIMPUT file with the source description, we will use XSPEC to specify the model parameters, and the SIXTE tool `simputfile` to build the SIMPUT file. Our source will have 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} (roughly the same spectral shape as the Crab nebula and a flux of 1 mCrab). We will also add two narrow Gaussian lines at energies of 6.39 keV and 6.41 keV.

```
computer:$ xspec

XSPEC version: 12.11.1c
Build Date/Time: Sat Oct 17 14:13:57 2020

XSPEC12>model phabs(pegpwrlw + gaussian + gaussian)

Input parameter value, delta, min, bot, top, and max values for ...
              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
            6.5       0.05(     0.065)          0          0      1e+06      1e+06
6:gaussian:LineE>6.39
            0.1       0.05(     0.001)          0          0         10         20
7:gaussian:Sigma>1e-3
              1       0.01(      0.01)          0          0      1e+20      1e+24
8:gaussian:norm>2e-4
            6.5       0.05(     0.065)          0          0      1e+06      1e+06
9:gaussian:LineE>6.41
            0.1       0.05(     0.001)          0          0         10         20
10:gaussian:Sigma>1e-3
              1       0.01(      0.01)          0          0      1e+20      1e+24
11:gaussian:norm>1e-4

========================================================================
Model phabs<1>(pegpwrlw<2> + gaussian<3> + gaussian<4>) 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
   6    3   gaussian   LineE      keV      6.39000      +/-   0.0
   7    3   gaussian   Sigma      keV      1.00000E-03  +/-   0.0
   8    3   gaussian   norm                2.00000E-04  +/-   0.0
   9    4   gaussian   LineE      keV      6.41000      +/-   0.0
  10    4   gaussian   Sigma      keV      1.00000E-03  +/-   0.0
  11    4   gaussian   norm                1.00000E-04  +/-   0.0
________________________________________________________________________

XSPEC12>save model mcrab_lines.xcm
XSPEC12>quit
```

Next, to generate the SIMPUT file with the `simputfile` tool, write a small shell script containing:

```
#!/bin/sh

base=mcrab_lines

$SIMPUT/bin/simputfile \
  Simput=${base}.simput \
  Src_Name=first \
  RA=0.0 Dec=0.0 \
  srcFlux=2.137e-11 \
  Elow=0.1 Eup=20 \
  Nbins=19900 \
  logEgrid=yes \
  Emin=2 Emax=10 \
  XSPECFile=${base}.xcm \
  clobber=yes
```

Save the script as `mcrab.bash`, make it executable (`chmod +x ./mcrab.fits`, and execute it.

2.1.2 Step 2: Run Simulation

We can now run observation simulations of this source with both instruments. For a microcalorimeter like Resolve, we need the `xifupipeline` tool, while a CCD like Xtend can be run using the `runsixt` tool. All the instrument properties are set in XML-files[1]. We note that as we simulate a point source, we explicitely only simulate on chip of Xtend.
Create a script with the following content:

```
#!/bin/bash

base=mcrab_lines
xmldir=$SIXTE/share/sixte/instruments/xrism

xifupipeline \
  XMLFile=${xmldir}/resolve/resolve_baseline_GVclosed.xml \
```

[1] In the future, we will have one tool `sixtesim`, which runs any detector type and only needs one XML file.

```
  AdvXml=${xmldir}/resolve/resolve_detector.xml \
  Simput=${base}.simput \
  EvtFile=output/evt_resolve.fits \
  RA=0.0 Dec=0.0 \
  Exposure=200000

runsixt \
  XMLFile=${xmldir}/xtend/xtend_ccd2.xml \
  Simput=${base}.simput \
  EvtFile=output/evt_xtend.fits \
  RA=0.0 Dec=0.0 \
  Exposure=100000
```

Make the script executable and execute it.

2.1.3 Step 3: Analyze Simulation

We can now analyze the event files produced during the simulation.

Image First, we generate an image using `imgev`. The main input is to give the WCS coordinate system for the detector. CDELT is given as `atan(pixel_pitch/focal_len)` in degrees. For CRPIX/CRVAL, in case of Resolve we currently assume that the focal point is in the exact center of the array and for Xtend the upper right edge of CCD2 is located 8 mm from the focal point in both DETX and DETY.

```
#!/bin/bash

$SIXTE/bin/imgev \
  EvtFile=output/evt_resolve.fits \
  Image=output/img_resolve.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=6 NAXIS2=6 \
  CUNIT1=deg CUNIT2=deg \
  CRVAL1=0.0 CRVAL2=0.0 \
  CRPIX1=3.5 CRPIX2=3.5 \
  CDELT1=-85.12516e-04 CDELT2=85.12516e-04 \
  history=true clobber=yes

$SIXTE/bin/imgev \
  EvtFile=output/evt_xtend.fits \
  Image=output/img_xtend.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=640 NAXIS2=640 \
  CUNIT1=deg CUNIT2=deg \
  CRVAL1=-0.0 CRVAL2=0.0 \
  CRPIX1=473.34 CRPIX2=473.34 \
  CDELT1=-4.9110668e-04 CDELT2=4.9110668e-04 \
  history=true clobber=yes
```

You can then look at the image with standard astronomical image viewers such as `ds9` or `fv`.

Spectrum To produce a spectrum, we can use the `makespec` tool:

```
#!/bin/bash

xmldir=$SIXTE/share/sixte/instruments/xrism

$SIXTE/bin/makespec \
  EvtFile=output/evt_resolve.fits \
  Spectrum=output/spec_resolve.pha \
  RSPPath=${xmldir}/resolve/ \
  clobber=yes

$SIXTE/bin/makespec \
  EvtFile=output/evt_xtend.fits \
  Spectrum=output/spec_xtend.pha \
  RSPPath=${xmldir}/xtend/ \
  clobber=yes
```

It is also possible to perform filtering here using `ds9` region files or the FITS extended filename syntax.

Grading For Resolve, the grading ratio of the measured events is also of large interest. This can be very easily extracted from the output event file, either by a (phython) script or directly from the command line with

```
fhisto infile=output/evt_resolve.fits outfile=output/grades.fits \
       column="GRADING" binsz=1 lowval=0 highval=4
```

Note that something similar can be done for pile-up in Xtend. Here the relevant information is stored in the header of the event file. The pile-up fraction defined by the ratio of the two entries `NPVALID/NVALID`, where `NPVALID` is the number of pile-up events and `NVALID` the number of valid detected events in the event file.

2.2 Bright Sources

For the second example, we will run Resolve simulations of our point source to investigate the dependence of the event grade distribution on the source brightness.

2.2.1 Step 1: Run Simulation

We will create a script that runs simulations of our point source with increasing source fluxes:

```
#!/bin/bash

xmldir=$SIXTE/share/sixte/instruments/xrism/resolve

for fluxfac in 10 50 100 500 1000
do
  # use FITS extended filename syntax to scale the source flux
  INFILE="mcrab.simput[SRC_CAT][col FLUX=FLUX*${fluxfac},*]"

  # scale exposure time inversely with flux to keep constant number
  # of photons
```

```
  EXP=$(python -c "print(20000/${fluxfac})")

  # run a simulation with the standard configuration
  xifupipeline \
    XMLFile=${xmldir}/resolve_baseline_GVclosed.xml \
    AdvXml=${xmldir}/resolve_detector.xml \
    Simput="${INFILE}" Exposure=${EXP} \
    EvtFile=output/evt_resolve_x${fluxfac}.fits \
    RA=0.0 Dec=0.0

  # and a simulation with an additional filter
  xifupipeline \
    XMLFile=${xmldir}/resolve_fwND_GVclosed.xml \
    AdvXml=${xmldir}/resolve_detector.xml \
    Simput="${INFILE}" Exposure=${EXP} \
    EvtFile=output/evt_resolve_fwND_x${fluxfac}.fits \
    RA=0.0 Dec=0.0
done
```

2.2.2 Step 2: Analyze Simulation

We now want to plot the grade distribution versus the source flux. One way to implement this would be with the help of the following Python script. Here we show the Resolve as an example, but a similar analysis can be done with pile-up in Xtend. In this case one would need the PILEUP column of the output event files file.

```
from astropy.io import fits
import numpy as np
import matplotlib.pyplot as plt
import sys

fluxfac = np.array([10,50,100,500,1000])
grades = np.arange(5)
gradenames = ["Hp","Mp","Ms","Lp","Ls"]

filters = ["", "fwND_"]

fig,ax = plt.subplots(len(filters))
for i_filt in range(len(filters)):

  # save grading information in a 2D array
  hists = np.zeros((len(fluxfac),len(grades)))

  filt = filters[i_filt]

  # loop over simulation output files
  for i_flux in range(len(fluxfac)):
    infile = "output/evt_resolve_{}x{}.fits".format(filt,fluxfac[i_flux])
```

```
    # for each event, grading information is stored in the "GRADING" column
    evtgrades = fits.open(infile)[1].data["GRADING"]

    # determine the distribution of grades for this observation
    dist,bins = np.histogram(evtgrades,np.arange(6))
    hists[i_flux,:] = dist /np.sum(dist)

  # plot the grade distribution
  for grad in grades:
    ax[i_filt].plot(fluxfac,hists[:,grad], label=gradenames[grad])

  # set labels etc.
  ax[i_filt].set_xscale("log")
  ax[i_filt].legend()

  if (len(filt)==0):
    ax[i_filt].text(fluxfac[-1],0.9,"No Filter",ha="right")
  else:
    ax[i_filt].text(fluxfac[-1],0.9,filt[:-1],ha="right")

  ax[i_filt].set_xlabel("Source flux [mcrab]")
  ax[i_filt].set_ylabel("Branching ratio")

plt.show()
```

2.3 First Light Image of N132D

The last example demonstrates a simulation of the first light image of N132D. The simulation calls are the same as before, just with a different SIMPUT file and pointing.

2.3.1 Step 1: Create SIMPUT

```
#!/bin/sh

SrcRA=81.259404
SrcDec=69.6437

$SIMPUT/bin/simputfile \
  Simput=n132d_flat.simput \
  Src_Name=n132d \
  RA=81.259404 Dec=69.6437 \
  Emin=0.3 Emax=8 \
  srcFlux=32e-11 \
  Nbins=19900 \
  logEgrid=n \
  Elow=0.1 Eup=15 \
  XSPECFile=plaw_shock.xcm \
  ImageFile=n132d_0.75-7keV.fits \
```

```
  clobber=yes
```

2.3.2 Step 2: Run Simulation

```
#!/bin/bash

xmldir=$SIXTE/share/sixte/instruments/xrism

SrcRA=81.259404
SrcDec=69.6437

xifupipeline \
  XMLFile=${xmldir}/resolve/resolve_baseline_GVclosed.xml \
  AdvXml=${xmldir}/resolve/resolve_detector.xml \
  Simput=n132d_flat.simput \
  EvtFile=output/evt_resolve.fits \
  RA=${SrcRA} Dec=${SrcDec} \
  Exposure=300000

# radec2xy adds WCS coordinates to an event file
radec2xy \
  EvtFile=output/evt_resolve.fits \
  RefRA=${SrcRA} RefDec=${SrcDec} Projection=SIN

runsixt \
  XMLFile=${xmldir}/xtend/xtend_ccd2.xml \
  Simput=n132d_flat.simput \
  EvtFile=output/evt_xtend.fits \
  RA=${SrcRA} Dec=${SrcDec} \
  Exposure=10000

radec2xy \
  EvtFile=output/evt_xtend.fits \
  RefRA=${SrcRA} RefDec=${SrcDec} Projection=SIN
```

2.3.3 Step 3: Analyze Simulation

Image

```
#!/bin/bash

SrcRA=81.259404
SrcDec=69.6437

$SIXTE/bin/imgev \
  EvtFile=output/evt_resolve.fits \
  Image=output/img_resolve.fits \
  CoordinateSystem=0 Projection=TAN \
```

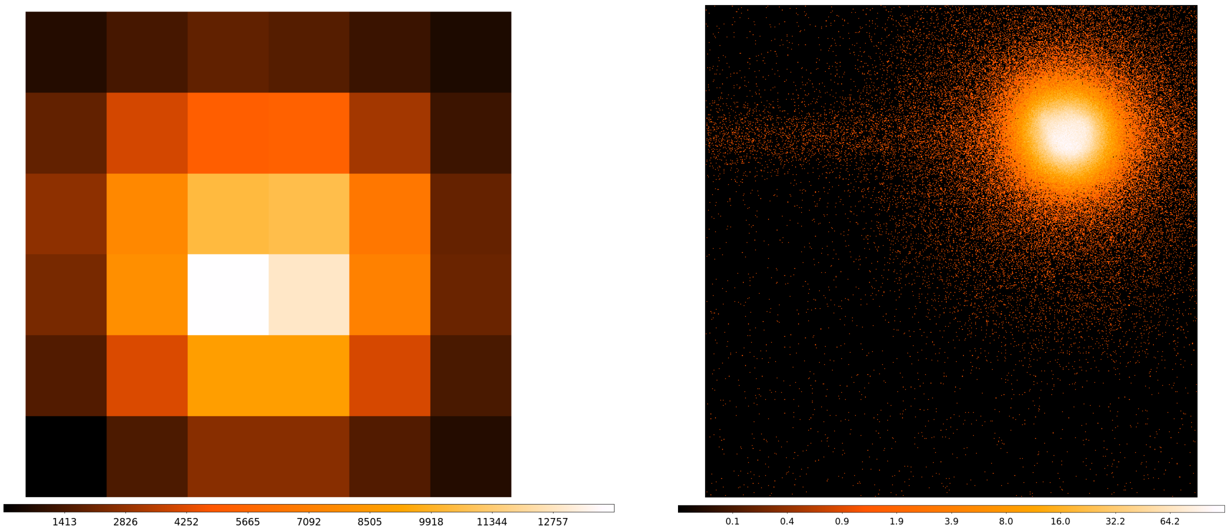

Figure 1: Images of the N132D simulations for Resolve (left) and Xtend (right).

```
  NAXIS1=6 NAXIS2=6 \
  CUNIT1=deg CUNIT2=deg \
  CRVAL1=${SrcRA} CRVAL2=${SrcDec} \
  CRPIX1=3.5 CRPIX2=3.5 \
  CDELT1=-85.12516e-04 CDELT2=85.12516e-04 \
  history=true clobber=yes

$SIXTE/bin/imgev \
  EvtFile=output/evt_xtend.fits \
  Image=output/img_xtend.fits \
  CoordinateSystem=0 Projection=TAN \
  NAXIS1=640 NAXIS2=640 \
  CUNIT1=deg CUNIT2=deg \
  CRVAL1=${SrcRA} CRVAL2=${SrcDec} \
  CRPIX1=473.34 CRPIX2=473.34 \
  CDELT1=-4.9110668e-04 CDELT2=4.9110668e-04  \
  history=true clobber=yes
```

Figure 1 shows the resulting images.

Spectrum

```
#!/bin/bash

xmldir=$SIXTE/share/sixte/instruments/xrism

$SIXTE/bin/makespec \
  EvtFile=output/evt_resolve.fits \
  Spectrum=output/spec_resolve.pha \
```

```
  RSPPath=${xmldir}/resolve \
  clobber=yes

$SIXTE/bin/makespec \
  EvtFile=output/evt_xtend.fits \
  Spectrum=output/spec_xtend.pha \
  RSPPath=${xmldir}/xtend \
  clobber=yes
```