eimsim

November 4, 2014

Abstract

Contains several tasks to help make simulated XMM-Newton x-ray images, to detect sources on them, then to assess the efficiency of the source detection.

1 Instruments/Modes

Instrument	Mode
EPIC	Imaging

2 Use

pipeline processing	no
interactive analysis	yes

3 Which documents to read

At last count, there are 25 tasks within the package **eimsim**, each with its own documentation (well, they will have, eventually). However, you don't need to read 25 sets of documentation! I recommend that you read the documentation for only 4 of the tasks, in the following order:

1. **eimsim** (ie, the present document)
2. **eimsimprep**
3. **eimsimbatch**
4. **eimsimreduce**

4 Description

4.1 Quick cookbook

For a minimum-fuss run of the simulation code, do as follows:

1. Create a subdirectory 'product' off your present working directory (PWD).
2. Put into this directory the following files:

   - A Calibration Index File (CIF);
   - An attitude history file;
   - calibrated event lists for one or more EPIC instruments;
   - for each of the instruments for which you have supplied an event list, vignetted exposure maps in 5 energy bands.

   These files are expected to have the standard XMM-SSC product filenames. Examples for EPIC MOS1 are as follows:

   ```
   P0123456789OBX000CALIND0000.FIT
   P0123456789OBX000ATTTSR0000.FIT
   P0123456789M1S002MIEVLI0000.FIT
   P0123456789M1S002EXPMAP1000.FIT
   P0123456789M1S002EXPMAP2000.FIT
   P0123456789M1S002EXPMAP3000.FIT
   P0123456789M1S002EXPMAP4000.FIT
   P0123456789M1S002EXPMAP5000.FIT
   ```

   Note that all these files are used merely as templates - that is, the only parts of them which are used by **eimsim** are those parts which give information about image exposure, plus the pixel grid location (from the WCS keywords) and dimensions. No data values are used which depend on real, observed x-ray events.

3. Set the SAS_CCFPATH environment variable to the location of a CCF set which you know to be valid for the version of the SAS which you are running.

4. Copy the source template dataset `srcspec_1xmm.fits` (or `srcspec_2xmm.fits`) from the directory `$SAS_DIR/lib/data/eimsimdata/` to your PWD and rename it '`srcspec.fits`'.
5. Run in succession the following commands:

```
eimsimprep
eimsimbatch
eimsimreduce
```

The files produced are described in the 'Output Files' sections in the documentation for these three tasks. Expect the middle script to take on the order of 10 minutes to run (the 1st and 3rd should be much quicker). If you want to understand exactly what these commands have done you will have to find out what the default values of all parameters were set to, either by looking in the <command>.par file or doing '<command> -h' or '<command> -d'.

4.2 Overview of the eimsim package.

The primary function of the **eimsim** package is to test procedures for detecting sources in images taken by the EPIC x-ray cameras on XMM-Newton. Simulated EPIC images are constructed; these images are subjected to a source-detection procedure; finally, the detection results are assayed.

4.2.1 What is and isn't simulated

Not all features found in real XMM-Newton EPIC images are reproduced in the simulated images. The following table summarises these differences:

Feature:	Simulated:
Poisson noise	yes
Off-axis PSF	yes
Chip gaps	yes
Vignetting	yes
Dead pixels (as expressed in the exposure maps)	yes
Background statistics	yes
Attitude wander (as expressed in the exposure maps)	yes
Bright pixels	no
Out-of-time events	no
Scatter from RGA	no
PN low-energy instrumental features	no
Pile up	no
Random source spectra	no
Extended sources	no

4.2.2 Relations between the components of the package

The sequence in which the **eimsim** tasks are called, and the command relations between them, can be diagrammed as follows:

- **eimsimprep**
  - **mosaicprep**
  - **padmask**
  - **bkgtemplategen**
  - A user-definable 'detprep' task. Examples provided are **eimsimdetprep1xmm** and **eimsimdetprep2xmm**.
- **eimsimbatch**
  - **eimsim** (see 'task' doco starting at section 4.3 below)
    - **srclistsim**
    - **newcolgen**
    - **srcmap**
    - A user-definable source-detection task. Examples provided are **eimsimdetect1xmm** and **eimsimdetect2xmm**.
      - Optionally **edetaux**
      - Optionally **eratetoflux**
    - **fluxlinearize**
    - **srccompare**
- **eimsimreduce**
  - **eimsimbias**
  - **eimsimcompleteness**
  - **eimsimreliability**

Other tasks from SAS packages other than **eimsim** are also called, as well as some ftools; but these 'foreign' calls have not been listed here.

As described in the cookbook (subsection 4.1 above), the minimum procedure still requires the user to run three separate tasks, **eimsimprep**, **eimsimbatch** and **eimsimreduce**. Why three - why can't all these functions be bundled into a single script? The answer is that the functionality has been divided between these three tasks to make it easier to run the simulations N times in order to generate an ensemble of statistically independent results. Some jobs need only to be done once: these are sequestered into **eimsimprep** or **eimsimreduce** as appropriate. Those jobs which need to be performed anew for each simulation run are performed by **eimsimbatch**. However, a glance at the **eimsimbatch** documentation will show that this task contains little more than a loop: at each iteration of the loop, **eimsim** is called. Thus **eimsim** performs the bulk of the simulation work, from generating a list of random sources to assaying the output of the detection procedure.

4.2.3 Input and output directories and many-observation mosaics:

The tasks in the **eimsim** package need to know the names of the directories in which they can respectively read input files and write output files. All the tasks construct these directory names in the same way: via parameters `obsidroots`, `prdssubdir`, `simgensubdir` and `simopsubdir`. (There is an additional 'directory' parameter pseudoprodsubdir which is intended just to contain non-vignetted exposure maps. If these ever become part of the standard product set, this parameter would become obsolete.) The parameter obsidroots is intended to contain a list of directory stem names (although at time of writing only 1 element is permitted). For each member of this list, the tasks append the string from parameter prdssubdir to the stem to generate the name of the directory in which to look for inputs. These input directories must be present before any **eimsim** tasks are run. The string from parameter simopsubdir is appended to the stem to generate the name of the directory in which to write observation-specific outputs; whereas simgensubdir gives rise to a single subdirectory off the PWD, which is to contain all the non-observation-specific output files. The **eimsim** tasks will 'mkdir' the appropriate simopsubdir and simgensubdir subdirectories if these are not already present.

As described in the cookbook subsection (4.1), the input files comprise templates taken from the SSC product files for a particular XMM observation. Each 'input' directory, derived from successive elements of the obsidroots list, must contain template files which relate to no more than a single XMM observation. Now, in the cookbook section, you will note that I specified that these files should be in a subdirectory named 'product' off the present working directory; the reason for this is that the default value of obsidroots is '.' and the default value of prdssubdir is 'product'.

To illustrate with an example in which some non-default values are used: The command

```
        eimsimprep obsidroots='/mydisk/obs1 /mydisk/otherfiles/obs2' \
          prdssubdir=infiles simopsubdir=sim_output simgensubdir=generic \
          pseudoprodsubdir=nonvig_files
```

generates the following subdirectory names:

```
Must pre-exist, and contain the proper products:
        /mydisk/obs1/infiles
        /mydisk/otherfiles/obs2/infiles

Created by the eimsim tasks:
        ./generic
        /mydisk/obs1/sim_output
        /mydisk/obs1/nonvig_files
        /mydisk/otherfiles/obs2/sim_output
        /mydisk/otherfiles/obs2/nonvig_files
```

The task will look for product templates from the first observation in /mydisk/obs1/infiles and in /mydisk/otherfiles/obs2/infiles for those from the second.

4.2.4 Multiple entry and exit points

All four of the central tasks (which are perl scripts) can be entered and exited at several places in the script. This behaviour is mediated via parameters entrystage and finalstage. The individual behaviours of these parameters are described more fully in the 'parameters' sections of the respective task documents.

4.2.5 Choice of source detection scheme

It is possible to run the simulation with different schemes for detecting sources in the simulated images. The task **eimsim** calls a separate script to perform the source detection. The user may either select one from the (small) range available, or supply their own. It is usually also necessary to supply a matching 'preparation' script to **eimsimprep**. Exact details of the necessary calls can be found in the documentation for these respective tasks. In 'cookbook' terms however, the two styles appear as follows:

- If the user chooses one of the supplied detection schemes, eg the 2xmm scheme, the 'cookbook' procedure changes to the following:
  1. Do items 1 to 3 from the 'cookbook' (section 4.1).
  2. Copy `$SAS_DIR/lib/data/eimsimdata/srcspec_2xmm.fits` into the PWD and rename it '`srcspec.fits`'.
  3. Run the tasks as follows:

     ```
     eimsimprep dettype=2xmm
     eimsimbatch
     eimsimreduce
     ```

- If the user wishes to supply their own detection and det-prep tasks, the 'cookbook' procedure changes to the following:
  1. Do items 1 to 3 from the 'cookbook' (section 4.1).
  2. Copy an appropriate template file from `$SAS_DIR/lib/data/eimsimdata/` into the PWD, or otherwise generate one; if you want to adhere to the present simple cookbook style you will then need to rename it 'srcspec.fits'.
  3. Run the tasks as follows:

     ```
     eimsimprep detpreptask=<name of my det prep task>
     eimsimbatch dettask=<name of my detection task>
     eimsimreduce
     ```

  Of course you can also use this command style to run one of the supplied detection schemes if you wish. If you do this you will find that there is some protection in place to prevent you running clashing preparation and detection scripts. You will also be warned if you try to use a 1xmm template set with 2xmm detection scheme, for example.

4.2.6 Parameter 'astest'

Most of the **eimsim** perl tasks have `astest` as a parameter. This offers a way to test the perl grammar without running any non-perl tasks.

4.2.7 Generic nature of many of the tasks

From general principles it seemed advisable to make as many of the sub-tasks as possible independent of XMM-Newton. These are tasks whose names don't start with an 'e'.

4.3 The task eimsim

It is not expected that the user will run this task directly: instead, the user should run **eimsimbatch**, which calls **eimsim** N times, where N is a user-specifiable number. All necessary **eimsim** parameters are also parameters of **eimsimbatch**, which passes them straight through to the subtask. **eimsimbatch** just contains a loop for performing the N calls, plus some machinery for cleaning up, and for 'nice' stopping. Thus it makes sense to place most of the detailed description of the simulation within the present document rather than in the documentation for **eimsimbatch**.

Task **eimsim** does 9 things in sequence, described in the following subsubsections:

4.3.1 Make list of simulated sources

This function may be performed alone by calling the script with `entrystage` and `finalstage`='makesimlist'.

The tasks **srclistsim** and **newcolgen** are called within this section. The first of these performs the bulk of the work of the section, which is to generate a list of random source positions and fluxes. The description of the probability distributions these sources are to follow is given in the template set pointed to by parameter `srcspecset`. This template file is a FITS dataset. Its structure is described in section 7: I'd advise you to print this section out and have it beside you to refer to as you read the remainder of the present section (4.3.1).

The explanation for the tables, keywords and columns in the template set is as follows. Table `SRCSPECS` contains information about the probability distributions of the source positions and fluxes; table `FLUX_SCALES` contains information about the source spectrum. (The template set contains other tables which are used in other parts of eimsim.)

The spatial locations of the sources are distributed evenly on the celestial sphere, but are restricted to a cone delimited by the `CONE_RA`, `CONE_DEC` and `CONE_RAD` keywords. The number density function $n(S)$ of source fluxes S is a piecewise power law

$$n(S) = k_i \; S^{-\gamma_i} \text{ for } S_i \leq S < S_{i+1}$$

such that the reverse-cumulative integral N of n is also a piecewise power law:

$$N(>S) = \frac{k_i}{\gamma_i - 1} S^{1-\gamma_i} \text{ for } S_i \leq S < S_{i+1},$$

where the units of N are sources deg^{-2}. For N to also be given by a piecewise power law it is necessary that $\gamma_i > 1$ for all i and that

$$\frac{k_i}{\gamma_i - 1} S_{i+1}^{1-\gamma_i} = \frac{k_{i+1}}{\gamma_{i+1} - 1} S_{i+1}^{1-\gamma_{i+1}}.$$

The log$N$-log$S$ of such a function appears like a sequence of m connected line segments. The function is defined in table `SRCSPECS` by the 'knee' points of this log$N$-log$S$. The ith row of the `FLUX` and `DENSITY` columns record S_i and $N(S_i)$ respectively. There must be $m \geq 1$ rows, since each row defines the left-hand anchor point of its line segment. The first row defines the faint-end cutoff of the log$N$-log$S$ - in other words, no source may be fainter than S_1. There is no need for a bright-end cutoff: it is assumed that the last power law continues to infinite flux. This means however that there is no uppermost 'knee' to anchor this last, mth power law. This mth line segment is defined instead by storing the value of $1 - \gamma_m$ in the `HI_SLOPE` keyword. A Euclidian distribution of sources in space would therefore result in a `HI_SLOPE` value of -1.5.

The `FLUX_SCALES` table records information relating to the source spectrum. At present, all the simulated sources are assigned the same spectrum. (This is not very realistic, and may be changed in future.) The spectrum is an absorbed power law, the photon spectral index being given by keyword `SPECINDX` and the HI column density by keyword `HI`. The keyword `FLUX` gives the total flux, within the band defined via the keywords `E_MIN` and `E_MAX`, from a source of such spectrum, in the case that the source flux density per unit energy is equal to 1 erg cm^{-2} s^{-1} keV^{-1} at 1 keV. The flux values of the simulated sources are given for this same band.

The **eimsim** package makes images in N energy bands. Exactly which bands are used is defined by the occurrence of the respective exposure-map templates (see section 7). In other words, if exposure maps are found only for bands 1, 2 and 4, then these are the only bands for which images will be made within **eimsim**. The `FLUX_SCALES` table must have a row for each of these bands. It may have additional rows for other bands, but these will be ignored by **eimsim**.

The first column of the `FLUX_SCALES` table records the ID integer of each tabulated energy band. Columns `E_LO` and `E_HI` define the band edges. Column `FLUX` records the flux in this band from a source of the spectrum defined by the keywords mentioned just above. Thus the flux of any simulated source in any of the N bands may be calculated by dividing the source flux by the keyword `FLUX` above, then multiplying by the appropriate value of the column `FLUX`.

The remaining columns of table `FLUX_SCALES` record the 'Energy Conversion Factors' or ECFs for each EPIC instrument and for four filters. These ECFs are used to convert fluxes S to count rates R as follows:

$$R = S \times ECF \times 10^{11}.$$

The output of the 'makesimlist' function of **eimsim** is a FITS dataset containing a binary table extension `SRCLIST` which has the following columns:

Name	Data type	Units
`INDEX`	4-byte int	
`FLUX`	4-byte real	erg cm^{-2} s^{-1}
`RA`	8-byte real	deg
`DEC`	8-byte real	deg
`FLUXRAND`	4-byte real	
`STREAM_N`	4-byte int	
`FIELD_N`	4-byte int	

Column `INDEX` is initially the same as the row number (starting at 1), but the `INDEX` value obviously follows the source if the list is filtered or sorted in any way. It offers therefore an unambiguous way to identify each source.

The `FLUX` value is the flux of the source within the band defined by the `E_MIN` and `E_MAX` keywords of table `FLUX_SCALES` in the template set, as described above.

Columns `RA` and `DEC` don't require any explanation.

The explanation of the `FLUXRAND` value is as follows. One well-known technique for generating random values of a coordinate x which have a probability distribution $P(x)$ is to integrate P then invert the result. That is, the random x values are generated from the formula

$$x = I^{-1}(y) \tag{1}$$

where y is an evenly-distributed random variable, I^{-1} denotes the inverse of I and

$$I(x) = \frac{\int_{-\infty}^{x} dx\, P(x)}{\int_{-\infty}^{\infty} dx\, P(x)}. \tag{2}$$

In the present case in which P and x are respectively the normalized differential sky density $n/n_{\max}$ and the flux S, the `FLUXRAND` value is the evenly-distributed random variable (y in equation 1) associated with the generated random flux value S (x in equation 1). This value is retained to assist in the matching of detected with simulated sources, as described in subsection 4.3.9.

Columns `STREAM_N` and `FIELD_N` are fully described in the **eimsimbatch** documentation, but briefly speaking, `STREAM_N` is created to allow **eimsim** to be run in parallel in different streams; whereas `FIELD_N` records the sequence number in the iteration performed by **eimsimbatch**.

A copy of the template table `SRCSPECS` is also appended to the source list.

4.3.2 Flag those sources outside the region of detectability

This function may be performed alone by calling the script with `entrystage` and `finalstage`='imsample'.

When the list of detected sources is matched with the list of simulated sources, it is undesirable to match with simulated sources which lie outside the region of sky where the exposure is significant. Such sources are, in essence, flagged at this stage by use of the task **imsample**. I say 'in essence' because what actually happens is that **imsample** measures the value of a reciprocal sensitivity map at each source position and writes this to a column `INV_SENSY`. The value of this column has no effect on the likelihood of a particular simulated source being matched with a detection, but sources which have a zero value in this column are screened out within the **eimsimcompleteness** and **eimsimreliability** tasks (called during script **eimsimreduce**).

4.3.3 Make a counts/pixel/sec image

This function may be performed alone by calling the script with `entrystage` and `finalstage`='makerateimg'.

The process of converting the list of source positions and fluxes into a realistic XMM-Newton image consists of three stages:

1. For each EPIC instrument, and within each band, make an ideal count-rate image (in theory of the whole sky, though in practice we don't bother with most of it). The units of this are (expected) counts per image pixel per second.
2. Multiply this by the exposure map to create an ideal event-count image. In other words, what this creates is a map of the expectation value for the event counts at each pixel.
3. Convert the expectation values to random, Poisson-distributed integer event counts.

The present function performs the first of these stages. Essentially this is done by adding an appropriate Point Spread Function (PSF) to the image for each sky location in the input source list. Ideally this would be done by reading a PSF from the XMM-Newton calibration data (the CCF). However in practice one finds that accessing this data is rather slow. Since there may be tens of thousands of simulated sources in the list, it is desirable to find some better procedure. For this reason it has been decided to use the CCF PSF only for those sources which are brighter than a user-settable flux level. For sources fainter than this level, the following axially-symmetric King function is used:

$$S(x-x_0, y-y_0) = S_0 \left(1 + \frac{[x-x_0]^2 + [y-y_0]^2}{\rho^2}\right)^{-\alpha}.$$

The parameters ρ and α are also settable by the user, but the default values, which are taken from values fitted to the on-axis PSF for the MOS camera (********** ref to RDS cal document XMM-SOC-CAL-TN-0018), should be perfectly adequate.

As said, the flux cutoff is a free parameter `fluxcutoff` of the **eimsim** script; the user is however advised to choose a value such that the proper, CCF PSF is used for all sources bright enough to be detectable. A few preliminary runs of **eimsim** may be required to determine a safe value.

Another point to consider is the size of the PSF 'patches' which are added to the image. There are two related issues:

1. How large should one make the patch array? 'Twice the size of the image' is the ideal answer from a theoretical standpoint, so that there would always be complete overlap between the patch and the image, no matter where on the image the centre of the PSF was located. On the other hand, practical issues to do with memory and computing time argue for as small an array as possible. Another consideration is the detection procedure - if a PSF fitting algorithm is employed as part of this, for example as in the sas task **emldetect**, then it is clear that the PSF patch used here within **eimsim** must be at least as large as that used within the fitting procedure. Failure to do this will result in bias in the fitted flux values, as seems to have been the case in previous versions of **eimsim**.
2. Should one vignet or 'feather' the PSF patch in order to avoid an abrupt drop to zero at the edges of the patch? This becomes particularly important if a small patch size is chosen. On the other hand, an unwise choice of feathering function will give rise to flux biases in any fitting procedure employed in the source detection (see above).

The PSF array size in **eimsim** is at present 'hard-wired' to the value of 21 by 21 pixels. The PSF parameters used for the faint sources result in a PSF which drops to about 9% of its central value at these array edges.

The feathering function chosen, let's call it y, is a piecewise inverted parabola, viz: $y = 1$ for $|x| < fs$,

$$y = 1 - \left(\frac{|x| - fs}{s(1 - f)} \right)^2$$

for $fs \leq |x| < s$, $y = 0$ else. Here x is the distance from the centre of the array, s is the array half size (=10 in the present case) and f (which should be between 0 and 1) is set somewhat arbitrarily to 0.7.

The effect of the feathering on the PSF is illustrated schematically in figures 1 to 3.

The bright sources are applied by use of the (XMM-specific) task **esrcmap**. This task obtains the PSF from the XMM calibration data via the **cal** library interface call CAL_getPSF, with the **cal** state variable accuracyLevel set to ACCURACY_MEDIUM. This returns a PSF which is interpolated from samples generated via a ray-tracing method. The source-detection task **emldetect** uses the same PSF by default.

The faint sources are added to the image by use of the non-XMM-specific equivalent task **srcmap**. This task uses the same PSF for each source, namely a King function. The relevant scale values are settable via **srcmap**, but not **eimsim**, parameters.

In order to preserve something close to realistic statistics in the background, it is usual to include, in the list of simulated sources, a large fraction which are too faint to be detected. This carpet of faint, confused sources supplies the desired 'lumpiness', but at the cost of raising the average level of vignetted background. The approximate amount of the added background (in terms of flux in the simulated source energy band per unit sky area) can be estimated either from the first plot created by **eimsimcompleteness**, or by use of **faintbackcalc**. Once the added flux per square arcsec is known, it can be subtracted from the source images by use of `eimsim` (and **srcmap**) parameters `withfluxoffset` and `fluxoffset`.

4.3.4 Make a counts/pixel expectation-value image

This function may be performed alone by calling the script with `entrystage` and `finalstage`=‘makectsimg’.

This performs the second of the three stages listed above. Essentially it just consists of multiplying each of the rate images obtained from the previous (first) stage by the appropriate exposure map. The background expectation-counts image which was made in **eimsimprep** is then added.

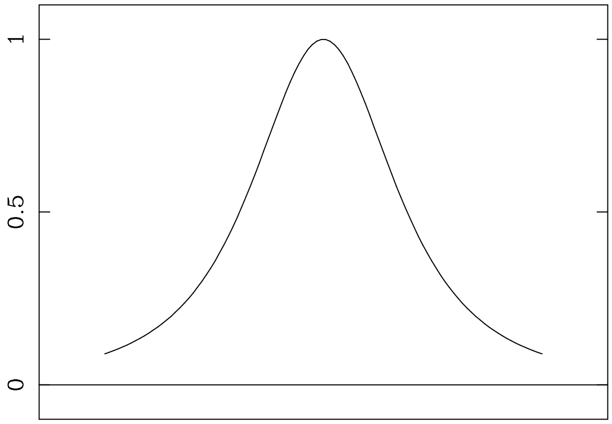

Figure 1: Section through the raw, truncated PSF.

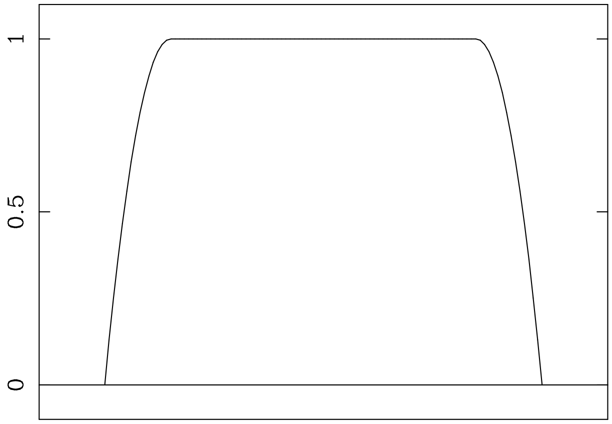

Figure 2: The vignetting or feathering function.

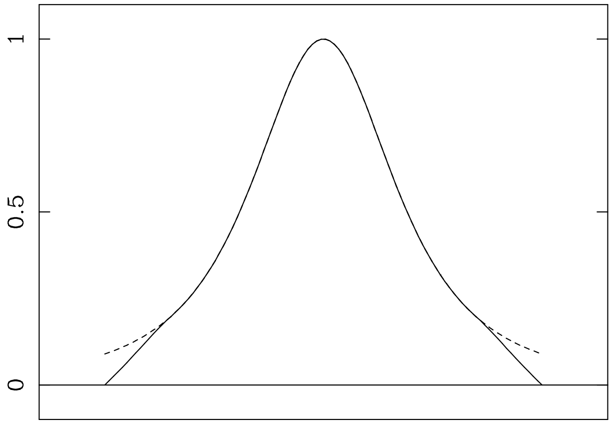

Figure 3: The PSF after feathering.

4.3.5 Make a random, Poisson-distributed events image

This function may be performed alone by calling the script with `entrystage` and `finalstage`='poissonize'. The randomisation is performed by the task **impoissonize**. The result is an integer-valued image which should look similar to a real XMM-Newton EPIC image of the given exposure.

4.3.6 Perform source detection

This function may be performed alone by calling the script with `entrystage` and `finalstage`='detect'.

This is the core of the simulation process: the detection of 'sources' in the simulated images. Exactly how this is performed is up to the user - ie the user can write their own detection script, and arrange very easily for this to be called by **eimsim**. In this way, several different schemes can be compared. At present, the **eimsim** package contains two example detection scripts: **eimsimdetect1xmm** and **eimsimdetect2xmm**, corresponding to the respective 'prep' scripts and template sets. The name of the detection script can be supplied to parameter `dettask` of **eimsim**, although a more robust way is available, as described in section 4.2.5 of the package documentation.

A user-written detection script must supply the detected sources in a FITS binary table called `SRCLIST`. There must be no more than 1 row per source. The table must contain columns as follows:

Name	Data type	Units
`SRC_NUM`	4-byte int	
`FLUX`	4-byte real	erg cm^{-2} s^{-1}
`FLUX_ERR`	4-byte real	erg cm^{-2} s^{-1}
`RA`	8-byte real	decimal deg
`DEC`	8-byte real	decimal deg
`RADEC_ERR`	4-byte real	arcseconds
`DETEC_PNULL`	4-byte real	

Column `SRC_NUM` is an positive-valued integer which is unique to the source. Note that, for 1xmm and 2xmm detection schemes, the name of the output source ID column is hard-wired into **emldetect** or **srcmatch**, whichever is relevant; `SRC_NUM` contains these identical values; the column is just given a new, common name for convenience. The 'original' column name is written to the keyword `ID_COL`.

Note that the `FLUX` and `FLUX_ERR` values should be valid for the same energy band as the simulated sources. This band is defined by the `E_MIN` and `E_MAX` keywords of the `FLUX_SCALES` table in the source template file.

Column `DETEC_PNULL` records the probability $P_{\rm null}$ that the source is detected by chance. At present, both **eimsimdetect1xmm** and **eimsimdetect2xmm** take, for each source, the value of `DET_ML` from the row with `ID_INST`==0 and `ID_BAND`==0, and process it as follows to get `DETEC_PNULL`:

```
  DETEC_PNULL = exp(-DET_ML).
```

Task **eimsim** calls the supplied detection script using the same command-line format as for sas tasks, viz:

```
<detection task> obsidroots='<list>' refband=2 <etc>
```

The full list of parameters supplied to this command line is as follows:

Parameter	Passed in from eimsim	Comment
obsidroots	yes	
refband	yes	
prdssubdir	yes	
simopsubdir	yes	
simgensubdir	yes	
astest	yes	
srcspecset	yes	
srclistset	no	Name of the output source list
streamnumber	yes	
idnumber	yes	
entrystage	no	passed in from eimsim parameter –detentrystage
finalstage	no	passed in from eimsim parameter –detfinalstage

In order to avoid cross-talk when running several simulations in parallel, any intermediate files written by the detection script should have names which contain both the stream and id numbers. In addition, it is helpful to construct the detection script so that it has an optional entry point named 'cleanup'. The function of this portion of the detection script should be just to delete all intermediate files. In other words, the detection script should be so designed, that it deletes all its intermediate files if invoked as follows:

```
<detection task> entrystage=cleanup
```

If you opt to use one of the supplied detection scripts instead of writing your own, PLEASE NOTE that the 1xmm and 2xmm detection schemes employed different energy band definitions and different versions of non-**eimsim** sas tasks. You can obtain the correct band scheme by making use of the relevant template file in `$SAS_DIR/lib/data/eimsimdata/`. The filenames have 1xmm or 2xmm in them so it is not hard to tell which to use. The non-**eimsim** sas task versions are as follows:

Task	1XMM version	2XMM version
emask	2.7	2.9
eboxdetect	4.13.1	4.15.2
emldetect	4.32.1	4.44.25
esplinemap	4.0.3	4.4
srcmatch	3.15.1	not used

The matching det prep tasks will check these versions and issue a warning if the incorrect ones are found.

4.3.7 Transform the flux coordinate

This function may be performed alone by calling the script with `entrystage` and `finalstage`='fluxtorand'.

The purpose of this function is to prepare for the matching stage (section 4.3.9). In section 4.3.1, it is described how, for each simulated source, the original random number which became transformed into the flux value for that source was retained in the column `FLUXRAND`. Actually this number can easily be recalculated from equation 2, if one has the flux value. (In this equation x represents the flux, $I(x)$ the desired evenly-distributed random number and $P(x)$ is the differential logN-logS curve.) For purposes of matching detected and simulated sources it is desirable to apply the same transform to the detected source flux values. This is done by the present function, which calls **eimsim** task **fluxlinearize**. The result, and its uncertainty, are written respectively to 4-byte-real columns `LINF` and `LINF_ERR` in the list of detected sources.

4.3.8 Add various bits and bobs

This function may be performed alone by calling the script with `entrystage` and `finalstage`='addbits'. Added to the list of detected sources are (i) a keyword `SKY_AREA` which records the total area in square degrees of the area of non-zero exposure; (ii) columns `STREAM_N` and `FIELD_N` which record the values passed into the `streamnumber` and `idnumber` parameters. For the latter columns, all rows have of course the same value: they become useful only when several lists are merged during processing by **eimsimreduce**.

4.3.9 Attempt to match detected and simulated sources

This function may be performed alone by calling the script with `entrystage` and `finalstage`='compare'. The actual processing is done by a task called **srccompare**.

In order to asses how well the source detection machinery performs, we need some way to (i) match every detection with a unique member of the list of simulated sources which is the most likely identification, and (ii) measure the probablity that the match arose by chance. The obvious answer to the first requirement seems to be to find that simulated source which is 'nearest' in both position and flux to the detected source. This intuition can be quantified by imagining that both simulated and detected sources are represented by points in an abstract 3-dimensional space in which the first two axes record the source position,[1] and the third records the source flux. Let us define a quantity R in this space by the equation

$$R^2 = \left(\frac{x_{\rm sim} - x_{\rm det}}{\sigma_x}\right)^2 + \left(\frac{y_{\rm sim} - y_{\rm det}}{\sigma_y}\right)^2 + \left(\frac{S_{\rm sim} - S_{\rm det}}{\sigma_S}\right)^2,$$

where x, y and S represent position and flux respectively. The σ quantities represent the uncertainties which were determined by the source-detection procedure. For each detected source, we define its 'matching simulated source' as the one which minimizes R for that detection. Let us denote this minimum value of R by $R_{\rm match}$. The probability can then be obtained as follows. First, consider the ellipsoidal surface defined by

$$R^2_{\rm match} = \left(\frac{x - x_{\rm det}}{\sigma_x}\right)^2 + \left(\frac{y - y_{\rm det}}{\sigma_y}\right)^2 + \left(\frac{S - S_{\rm det}}{\sigma_S}\right)^2.$$

From the definition of $R_{\rm match}$, this ellipsoid has the following properties:

- It is centred on the 'position' in this abstract 3-dimensional space of the detected source.
- The principle axes of the ellipse preserve the ratios between the uncertainties. Indeed one can visualize the process of searching for a match as 'inflating' the ellipsoid as one inflates a balloon, until its edge intersects a simulated source.
- The ellipsoid just touches the matching simulated source.
- No other simulated source is found inside it.

[1] The RAs and decs are projected onto a plane tangent to the celestial sphere at the reference direction recorded in the WCS keywords `CRVAL1` and `CRVAL2` of the original template exposure maps. Use of such planar coordinates for x and y forces the $x - y - S$ space to be cartesian, in which it is straightforward to measure distances.

Intuition suggests that the larger the ellipse, or the larger the value of $R_{\rm match}$, the less likely it is that the detection is 'genuine'. Again we quantify this intuition by integrating the probability density distribution of simulated sources in position and flux over the ellipsoidal volume to give η, the expectation value for the number of simulated sources which would fall inside the ellipsoid by chance. Ok, we said above that there are zero sim sources within the ellipsoid - but that was in a single, particular case. What we want to test now is the null hypothesis, ie to ask how many simulated sources, on average, we would expect to land inside our ellipsoid if we threw the chips at random.

Having calculated η, it is fairly easy to see that the probability $P_{\rm null}$ of the null hypothesis is given by

$$P_{\rm null} = 1 - exp(-\eta). \tag{3}$$

There is a slight issue here, in that the simulated sources are not evenly distributed in S: the number of sources per flux interval increases greatly at low flux. This leads to a bias towards matching with fainter sources. In previous versions of **eimsim** I assumed that this was a bad thing, and took steps to transform the flux coordinate to correct for this. This is the point of the `FLUXRAND` business described in section 4.3.1. Now I am no longer sure that this is the case. In real life, we expect the gradient of number density with flux to bias the detected flux - this is called Eddington bias. Maintaining this bias during the matching stage ought to help correct for this. What concerns me more now is that the + and - flux uncertainties ought not to be the same in a simple flux scale: one would expect that the + one ought to be larger. Perhaps then the correct way to transform the flux scale before matching is to take its square root, which should even up the uncertainties. What I have done is provide the facility in **eimsim** to do any one of three things, namely (i) leave the flux alone; (ii) transform it to the `FLUXRAND` scale, in which the simulated sources are evenly distributed; (iii) transform the flux scale by taking square roots of flux. Comparison of empirical results ought to show which is the best procedure.

The following additional columns are written to the list of detected sources:

Name	Data type	Units	Comment
`X`	4-byte real	arcsec	X-coordinate of det source.
`Y`	4-byte real	arcsec	Y-coordinate of det source.
`X_ERR`	4-byte real	arcsec	X-coordinate error of det source.
`Y_ERR`	4-byte real	arcsec	Y-coordinate error of det source.
`SIM_X`	4-byte real	arcsec	X-coordinate of matching sim source.
`SIM_Y`	4-byte real	arcsec	Y-coordinate of matching sim source.
`SIM_FLUX`	4-byte real	erg cm^{-2} s^{-1}	Flux of matching sim source.
`SIM_INDX`	4-byte int		From simlist column `INDEX`.
`SIM_INV_SENSY`	4-byte real		From simlist column `INV_SENSY`.
`R_SIGMAS`	4-byte real		$R_{\rm match}$.
`MATCH_PNULL`	4-byte real		$P_{\rm null}$ from equation 3.
`SIM_LINF`	4-byte real		From simlist column `FLUXRAND`.
`FLAG`	4-byte int		

If the user chooses to take the square root of the flux coordinate then the following additional columns are written:

ROOTF	4-byte real	Square root of det source FLUX.
ROOTF_ERR	4-byte real	The appropriate error in L.
SIM_ROOTF	4-byte real	Square root of sim source FLUX.

The FLAG column is hardly used at present, but may be found useful in further analysis. Only bit 0 is set by task srccompare. If the same simulated source is 'claimed' by more than one detected source, bit 0 of the flag column is set for all the claimants except that with the smallest value of MATCH_PNULL.

This section also writes a keyword COMPARED='T' to the table header.

5 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints

obsidroots	no	string	.	

A list of directory names. For each member of the list, the task constructs subdirectory names by appending '/' followed respectively by the strings in prdssubdir and simopsubdir. Please see the respective parameter descriptions for further information.

entrystage	no	string	makesimlist	makesimlist-imsample-makerateimg-makectsimg-poissonize-detect-fluxtorand-addbits-compare-cleanup

This allows the user to enter the **eimsim** script at one of several places in its processing sequence.

finalstage	no	string	compare	makesimlist-imsample-makerateimg-makectsimg-poissonize-detect-fluxtorand-addbits-compare-cleanup

This allows the user to exit the **eimsim** script at one of several places in its processing sequence.

refband	no	string	1	

Where only one file is required out of a set spanning several energy bands, for example an exposure map to be used to create a detection mask, the band used is specified by this parameter.

prdssubdir	no	string	product	

For each member of the list obsidroots, the task constructs subdirectory names by appending '/' followed by the string in prdssubdir. The task expects to find input files in the prdssubdir subdirectory, namely a set of template files in the form of XMM products (see section 7 for a detailed description). Product templates from only 1 observation may be present in any one prdssubdir subdirectory.

simopsubdir	no	string	sim_output	

For each member of the list obsidroots, the task constructs subdirectory names by appending '/' followed by the string in `simopsubdir`. The task writes observation-specific outputs to this directory.

simgensubdir	no	string	sim_generic	

The task writes non-observation-specific output to this directory.

streamnumber	no	int	1	

See the **eimsimbatch** documentation for a description of this.

idnumber	no	int	1	

See the **eimsimbatch** documentation for a description of this.

srcspecset	no	dataset	srcspec.fits	

This is the name of a FITS dataset which contains specification of the source probability distributions and also band-related specifications. See section 7 for a detailed description. Example files can be found in `$SAS_DIR/lib/data/eimsimdata/` .

withsimsources	no	bool	yes	

If the user sets this to ‘no’, images with no sources will be created. This provides a way to assess the number of false detections.

energyfraction	no	real	0.95	

This parameter is read only if `withsimsources` is set to ‘yes’. The value is piped through to the parameter of the same name of task **esrcmap**.

fluxcutoff	no	real	2.0e-15	

This parameter is read only if `withsimsources` is set to ‘yes’. For simulated sources which have fluxes above this value, **esrcmap** is called to add an XMM point spread function (PSF) to the simulated image; for sources fainter than this cutoff, **srcmap** (no ‘e’) is called, which employs a rotationally-symmetric King function as PSF.

withfluxoffset	no	bool	no	

This parameter is read only if `withsimsources` is set to ‘yes’. If `withfluxoffset`=‘yes’, `fluxoffset` is subtracted from each pixel of the output image of **srcmap**. The reason for this is to allow the user to compensate for the increase in vignetted background caused by the accumulation of numerous too-faint-to-be-detectable point sources.

fluxoffset	no	real	0	

This parameter is read only if both `withsimsources` and `withfluxoffset` are set to ‘yes’.

dettaskstyle	no	string	auto	user—auto

The user is able to specify which source detection program or script should be used by the simulations. The user may either choose between alternatives provided in the package, or write their own source-detection script. Matters are somewhat complicated by the need to also provide a ‘detection preparation’ script to **eimsimprep**. The preparation and detection scripts must match. Some guidance for coordinating these choices at package level is given in section 4.2.5. If `dettaskstyle` is left at its default of ‘auto’, **eimsim** attempts to read the detection style code from a file in the PWD called ‘eimsim_config’. This config file is written by **eimsimprep**. Failure to find the file, or non-recognition of the style code, will both generate an error. Otherwise the style code is translated into the name of the detection task to be used. If `dettaskstyle` is set to ‘user’, the task reads the name of the detection task directly from the parameter `dettask`.

dettask	no	string	eimsimdetect1xmm	

This parameter is read if `dettaskstyle` is set to ‘user’. It gives the name of the task or script which is to perform the source detection. The user may either choose between alternatives provided in the package, or write their own source-detection script. For users wishing to ‘roll their own’, a description of the ‘handshaking’ required of such a task is given in section 4.3.6.

withdetentrystage	no	bool	no	

If ‘yes’, the task looks for parameter `detentrystage`.

detentrystage	yes	string		

A parameter to allow the user to enter the detection script (if that’s what it is) at one of several points in its processing sequence.

withdetfinalstage	no	bool	no	

If ‘yes’, the task looks for parameter `withdetfinalstage`.

detfinalstage	yes	string		

A parameter to allow the user to exit the detection script (if that’s what it is) at one of several points in its processing sequence.

astest	no	bool	no	

If ‘yes’, no tasks are called except the source-detection script, which is also passed this parameter value to allow it to do the same.

6 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

label *(error)*
explanation

label *(warning)*
explanation
corrective action: this is the corrective action

7 Input Files

1. A FITS template file, as described in item 1 of the ‘input files’ section of the **eimsimprep** documentation, but with the following additional keywords in the `SRCSPECS` table header:
   - String keyword `FIELD_ID` (‘CONE’ is currently the only accepted value).
   - 8-byte real keyword `CONE_RA` in deg.
   - 8-byte real keyword `CONE_DEC` in deg.
   - 8-byte real keyword `CONE_RAD` in arcsec.
   - 8-byte real keyword `SKY_AREA` in deg^2.
2. Various XMM product files, as described in item 2 of the ‘input files’ section of the **eimsimprep** documentation.

3. A single map which is supposed to be a mosaic of all reciprocal-sensitivity maps relevant to the simulation in hand. (In fact at present the exposure maps at the refband are used instead of reciprocal-sensitivity maps.) The image header must also contain the keyword SKY_AREA. This file is created by **eimsimprep**.
4. For each observation, instrument, exposure and band: a mask image dataset which specifies those parts of the sky on which simulated sources are to be placed. These are created by **eimsimprep**.
5. For each observation, instrument, exposure and band: a real-valued image of the expectation value, in average counts per image pixel, of the instrumental background. These are created by **eimsimprep**.

8 Output Files

1. A FITS dataset which contains a list of simulated sources in a table named SRCLIST. This file is written to the simgensubdir subdirectory. The SRCLIST table contains the following columns:
   - INDEX
   - FLUX
   - RA
   - DEC
   - FLUXRAND
   - STREAM_N
   - FIELD_N
   - INV_SENSY
   - DET_SRC_ID

   All these except for the last are described in section 4.3.1. The last column is added during the comparison process (section 4.3.9).
2. A FITS dataset which contains a list of detected sources in a table named SRCLIST. This file is written to the simgensubdir subdirectory. The SRCLIST table contains the following columns:
   - Described in section 4.3.6:
     - String keyword ID_COL
     - SRC_NUM
     - FLUX
     - FLUX_ERR
     - RA
     - DEC
     - RADEC_ERR
     - DETEC_PNULL
   - Described in section 4.3.7:
     - LINF
     - LINF_ERR
   - Described in section 4.3.8:
     - STREAM_N

  - FIELD_N
- Described in section 4.3.9:
  - X
  - Y
  - X_ERR
  - Y_ERR
  - SIM_X
  - SIM_Y
  - SIM_FLUX
  - SIM_INDX
  - SIM_INV_SENSY
  - R_SIGMAS
  - MATCH_PNULL
  - SIM_LINF
  - FLAG
  - (Optional) ROOTF
  - (Optional) ROOTF_ERR
  - (Optional) SIM_ROOTF

References