rgsrmfgen

June 2, 2019

Abstract

Creates a response matrix file for use with XSPEC and **rgsfluxer**.

1 Instruments/Modes

Instrument	Mode
RGS	Spectroscopy
RGS	High Time Resolution

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

rgsrmfgen creates an OGIP-compliant response matrix file, or files, as required for spectrum analysis within XSPEC. Because a certain level of compatibility between the spectrum and response matrix is required, it is typical and convenient (but not required) to create the spectrum first and provide it as in input to the response generator. The instrument's response within each incident (photon) energy bin is calculated for every dispersion channel. The size of the matrix is determined, therefore, by the size of the spectrum and the number of incident energy bins (`rows`) specified on the command line. A matrix of one hundred energy bins can be computed fairly quickly (about ten minutes) and is good for visual inspection, but is not good for analysis. Two hundred fifty bins is a minimum for use with **rgsfluxer**, and three thousand is probably acceptable for use with XSPEC. Be prepared to wait a while, and for periodic progress updates set the verbosity level to five.

The new parameter `witharffile` can be used to create two separated redistribution matrix (RMFMAT) and effective area (ARFMAT) files. The user has to make sure that the task **rgsfluxer** is used in one of the follwoing ways:

- Run **rgsfluxer** with redistribution matrix which contains the effective area (RSPMAT) file.

- Run **rgsfluxer** with redistribution matrix (RMFMAT) and the effecitve area (ARFMAT) file.

The metatask **rgsproc** handles this situation automatically and run **rgsfluxer** with the correct intput files depending of the user's choice.

If the `dyneffareacorr` is enabled, the **cal** calculates a correction to the effective area which is based on a Chebyshev polynomial parametrization of calibration data. This effective area correction is time dependent and around every six months, a new correction is added to CCF. Depending on the observation date, the **cal** interpolates between the two closest effective area correction epochs. If the observation date is later than the last available correction epoch, we use the latest one as the valid correction. If the observation date is earlier than the first available correction epoch, we use the first one.

The EFFAREACOOR CCF file has a new extension called RECTIFICATION that can optionally be applied using the `withrectification` switch.

4 Parameters

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

Parameter	Mand	Type	Default	Constraints

In general for this task only one filename need be provided and the rest are inferred by default using the provided filename as a template. The rules governing this behavior are intended to follow what the user would instinctively expect and as a result are somewhat complicated. The descriptions below focus on the meaning of each filename parameter, leaving the exact details of the inferred default value to be explained in the algorithm section.

evlist	no	dataset		

The filtered event list. It must contain the `EVENTS` table and exposure map extensions. Not modified. The default value infers the EVENLI file.

srclist	no	dataset		

The source description list. It must contain the `SOURCES` table and the relevant selection region extensions. Not modified. The default value infers the SRCLI_ file.

rmfset	no	file		

The response matrix file (OGIP-compliant format). Either created or modified depending on parameter `newrmf`. When modifying an existing response matrix (`newrmf`=no) an explicit value for this filename must be provided, which then becomes the template from which other filenames may be inferred. For a new response matrix the default value infers the RSPMAT file.

witharffile	no	boolean	no	

Switch to enable the creation of two independent response files ARF and RMF.

arfset	no	file		

The ARF file name (OGUP-compliant format).

emax	no	real	2.8	non-negative

When creating a new response matrix (`newrmf`=yes), this is the upper edge of the highest incident-energy bin to be included. When overwriting part of an existing matrix (`newrmf`=no), this identifies the upper incident-energy bin in the interval to be recomputed. This value will be raised automatically as necessary to maintain the contiguity of the bins, and can be used to extend the range of the existing matrix.

emin	no	real	0.3	non-negative

When creating a new response matrix (`newrmf`=yes), this is the lower edge of the lowest incident-energy bin to be included. When overwriting part of an existing matrix (`newrmf`=no), this identifies the lower incident-energy bin in the interval to be recomputed. This value will be lowered automatically as necessary to maintain the contiguity of the bins, and can be used to extend the range of the existing matrix.

rows	no	integer	4000	positive

The number of incident-energy bins to cover the range from `emin` to `emax`. Typically, when overwriting part of an existing matrix (`newrmf`=no), the intention is to increase the resolution of the matrix over a small interval. This can yield a matrix of manageable size that has very high resolution in just the right places.

fftdim	no	integer	3	1-5

The wings of the various distributions that are convolved together to form the narrow features of the line-spread function are truncated to limit the size of the convolution space. An increment of one in this parameter doubles the size of the convolution space; the larger the convolution space, the slower the computation and the less power is lost to the truncation.

newrmf	no	boolean	yes	

Whether to create a new response matrix or edit an existing one.

The remaining parameters apply only when `newrmf`=yes. When editing an existing matrix these specifications are obtained from its contents.

withspectrum	no	boolean	yes	

Determines how various details of the response matrix are specified. On true: enables parameter `spectrumset`, and the response matrix is made compatible with this spectrum with respect to its source, order, background correction, and channel definitions. On false: enables parameters `source`, `order`, and `bkgcorrect`, while the channel definitions are obtained from the event list (parameter `evlist`). Note

that a channel rebinning factor cannot be specified except through the `spectrumset`, an oversight which may be corrected in a future version.

spectrumset	no	dataset	srspec.ds	

A spectrum file produced by **rgsspectrum** from the data in the given event list (parameter `evlist`). Not modified. The metadata from this file are used to ensure that the response matrix produced is the suitable companion to this spectrum for the purpose of analysis. A default filename cannot be inferred; when this parameter is enabled (`withspectrum`=yes) an explicit value must be provided, which then becomes the template from which other filenames may be inferred.

source	no	integer	0	non-negative

If a spectrum file is not provided (`withspectrum`=no) this parameter specifies the `INDEX` number of a source in the source description list (parameter `srclist`), and thereby the incidence angle and selection regions for the response matrix. The default value, zero, indicates the `SOURCEID` attribute of the `EVENTS` table in the event list (parameter `evlist`), which is the source that was used to perform the aspect-drift corrections.

order	no	integer	1	positive

If a spectrum file is not provided (`withspectrum`=no) this parameter specifies the reflection order for computing the response matrix. In conjunction with the `source` parameter, this identifies the energy selection region from the source description list (parameter `srclist`). Although it does include the contribution of the LSF for every order, the final response matrix is nevertheless order-specific with respect to its effective area and `EBOUNDS` extension.

bkgcorrect	no	boolean	yes	

If a spectrum file is not provided (`withspectrum`=no) this parameter specifies whether background correction should be included in the response matrix computation. If yes, then the background selection region from the source description list (parameter `srclist`) is required. *In HTR mode the background correction to the response matrix is not defined, and this parameter is quietly ignored.*

withmirrorpsf	no	boolean	yes	

Whether to convolve the standard mirror PSF distribution into the LSF. *This option should be used with care and knowledge.* An appropriate circumstance for omitting the mirror PSF is when a custom angular distribution is provided (`withangdist`=yes) that empirically establishes the PSF contribution.

withangdist	no	boolean	no	

Whether to convolve a custom angular distribution into the LSF. *This option should be used with care and knowledge.* Enables parameter `angdistset`. Note that this distribution is one-dimensional, an arbitrary function of dispersion off-axis angle. It is rebinned onto the dispersion channel space according to the order and central incident energy of each LSF. The problem of reducing a two-dimensional image to this one-dimensional distribution is left to the user.

angdistset	no	file	angdist.txt	

An ASCII file containing a one-dimensional angular distribution along the axis of dispersion in arc minutes. Not modified. Each line of the file describes a distribution bin in terms of the start and stop angles, and the value assigned to that bin. The distribution may be normalized to any value, though unity is expected and recommended. See the input files descriptions for further details.

dyneffarecorr	no	boolean	yes	

Enable the dynamic effective area correction based on Chebyshev polynomials.

withrectification	no	boolean	no	

Use empirical RGS effective area correction

witheffectiveareacorrection	no	boolean	no	

Use RGS effective area correction

spectrumbinning	no	choice	lambda	lambda beta

Dispersion binning type.

5 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.

NullFileName *(error)*
A required filename parameter has been left empty, indicating that a runtime default value should be supplied by the task, but an appropriate default cannot be inferred from the other parameter values given.

FileNotFound *(error)*
The angular distribution file (parameter `angdistset`) could not be accessed.

badSource *(error)*
The `SOURCES` table has no entry with the specified source index number (parameter `source`).

nullExposure *(error)*
Either the event list (parameter `evlist`) contains no exposure map extensions at all, or none of the exposure maps enclose a non-zero area.

InvalidSpectrum *(error)*
This task is not designed to generate a response matrix consistent with a background spectrum (`HDUCLAS2`=BKG). If enabled, parameter `spectrumset` should refer to either a NET or a TOTAL spectrum.

InvalidRebinning *(error)*
A spectrum file has been provided (parameter `spectrumset`) and its channel definitions are incommensurate with the channels defined by the event list (parameter `evlist`). The spectrum channels must align with some integral rebinning of the `EVENTS` table dispersion channels.

InvalidMatrix *(error)*
The existing matrix to be modified has its rows (incident energy bins) in an unexpected order. This task requires that the incident energy bins be contiguous and in ascending order. Matrices not compliant with this requirement are not invalid according to the OGIP standard, but this task simply lacks the logic necessary to handle irregularities. In any case this task does not expect to be used to modify a response matrix generated by some other task, and so this error should never occur.

invalidMap *(warning)*
The event list provided (parameter `evlist`) contains an internal inconsistency: the channel definitions in the `EVENTS` table do not align with the channel definitions in the exposure map extensions.
corrective action: Each such exposure map is discarded, and no response will be associated with its node.

6 Input Files

- The event list. Must contain the `EVENTS` table and at least one exposure map extension, as produced by **rgsfilter**.
- The source description list. Must contain the `SOURCES` table and all applicable selection region extensions, as produced by **rgsregions**.
- Optional, a spectrum file. Must be of type NET or TOTAL, as produced by **rgsspectrum**.
- Optional, an angular distribution. Format: three-column ASCII, all values in floating-point representation. Non-conforming lines are discarded without error or warning. Each line describes a bin: starting angle, ending angle, bin value. The lines should be sorted in ascending order of the angles (otherwise the behavior is undefined). The angles are the dispersion off-axis coordinate in arc minutes, relative to the target source of the response matrix. This coordinate is similar to the `DELTA_DISP` column of the `SOURCES` table, except that `DELTA_DISP` includes the instrument-dependent F/L factor. This coordinate is also similar to the instrument-independent CAL θ, but follows the opposite sign convention. The author apologizes, and hopes that the benefits of this coordinate system justify the confusion it is bound to cause. Truncate the distribution at a reasonable width to avoid slowing the computation.

7 Output Files

The output file is an OGIP-compliant Response Matrix File (RMF) with the following extension tables. Only the non-standard elements are described here.

- MATRIX

  Attribute `LO_THRES` documents the value of the response threshold used when writing this extension. Channels with this value or less are considered empty and do not occupy space in the output. At present the user has no control over this threshold.

- EBOUNDS
- ANGULAR_DIST

  This extension is present only if the file was created with `withangdist`=yes.

`PHI0 PHI1`	real32	arcmin, dispersion angle bin boundaries
`VALUE`	real32	distribution value at this bin

8 Algorithm

- The following algorithm defines how certain filenames are inferred when not explicitly provided. This mechanism will work nicely when the task is run with product files from the PPS or from **rgsproc**. Note also that when the template filename is shorter than the PPS convention allows, an alternate convention is assumed, which the user may find more convenient under special circumstances.

```
IF *newrmf
  IF *withspectrum
    template = *spectrumset
  ELSE
    template = *evlist OR *srclist
ELSE
  template = *rmfset

IF *srclist == ""
  IF template.root.length >= 10
    *srclist = template with "SRCLI_0000" substituted
  ELSE
    *srclist = "srcli." + template.suffix

IF *evlist == ""
  IF template.root.length >= 10
    *evlist = template with "EVENLI0000" substituted
  ELSE
    *evlist = "evenli." + template.suffix

IF *newrmf AND *rmfset == ""
  IF template.root.length >= 10
    *rmfset = template with ("RSPMAT",*order,*source) substituted
  ELSE
    *rmfset = "rspmat." + template.suffix
```

- NarrowLSF

  The following algorithm defines the narrow-feature line-spread-function for a given source position, reflection order, and incident energy bin. The large-angle-scattering distribution is not included here because, in principle at least, it could be computed on a much coarser channel grid and thus consume much less space in the output. However, OGIP does not currently support a response matrix file format with both a fine-grid matrix and a coarse-grid matrix, and so the broad and narrow line-spread-functions are currently just added together as the response matrix is assembled.

```
IF GratingDataServer::beta(source,order,energy) within matrix channel space

  // prepare various distributions in wrap-around order for convolution
```

```
    // small angle scattering distribution
      D = GratingDataServer::scatteringDistribution(source,order,energy)
      (a,b) = GratingDataServer::scatteringDistributionContribs
      D[peak] += (1-a)*(1-b)

    // mirror point spread distribution
      IF *withmirrorpsf
        PsfDataServer::dispersionFigureDistribution(source,order,energy)

    // custom angular distribution
      IF *withangdist
        customFigureDistribution(order,energy)

    // grating bow induced misalignment distribution
      GratingDataServer::bowingFigureDistribution(source,order,energy)

    // geometric defocus approximation distribution
      GeometryDataServer::lsfDefocusDistribution(source,order,energy)

    // finite energy band broadening distribution
      GratingDataServer::broadeningDistribution(source,order,energy)

  // prepare the primary distribution on the matrix channel grid

    GratingDataServer::misalignmentFigureDistribution(source,order,energy)

  LSF = convolution(primary,various...)

ELSE

  // distributions peak outside the matrix channel space, so skip
  // convolutions, and instead just use the tail of the small angle
  // scattering distribution where it overlaps the channel space

    LSF = GratingDataServer::scatteringDistribution(source,order,energy)

// scale the LSF by the effective area

  IF dyneffarecorr
    scale = EffectiveAreaDataServer::realisticEffectiveAreaCurve(source,order)::area(energy)
  ELSE
    scale = EffectiveAreaDataServer::intrinsicEffectiveAreaCurve(source,order)::area(energy)

  IF withrectification
    scale = EffectiveAreaDataServer::effectiveAreaRectification(scale)::area(energy)

  NarrowLSF = scale * LSF
```

- BroadLSF
  The broad-feature line-spread-function is uncomplicated.

```
// place the large angle scattering distribution on the matrix channel
// grid and scale it by the effective area

  LSF = GratingDataServer::scatteringDistribution(source,order,energy)
```

```
scale = EffectiveAreaDataServer::intrinsicEffectiveAreaCurve(source,order)::area(energy)
BroadLSF = scale * LSF
```

- empiricalCrossLA
  The following calibration formula should eventually be transfered to the CAL. Input parameters are: a one-dimensional selection region in cross-dispersion channels, the source position, and the incident energy in keV.

```
dpsi = source.DELTA_XDSP * pi/10800
k = (energy * 10800) / (0.678 * pi)

FOREACH span = interval [min,max] of cross-dispersion channels
  s += (atan(k*(xdsp(max+0.5)+dpsi)) - atan(k*(xdsp(min-0.5)+dpsi))) / pi
  n += span.length

empiricalCrossLA = s/n
```

- The above two line-spread-functions are computed for each reflection order, including the zeroth, and combined with various response-suppressing factors to produce the output response matrix. The following algorithm shows the rough details of these calculations. Here the arrays prefixed with src and bkg refer, respectively, to the source and background spatial selection regions. In particular the arrays src.spans and bkg.spans are the selection regions themselves, converted to an image mask, and represented as a list of cross-dispersion channel intervals at each dispersion channel. Similarly the energy selection region for the reflection order of the output matrix is converted to the array called banana.

```
// analyze the exposure maps within the spatial selection regions

FOREACH node
  FOREACH b = dispersion channel of node
    src.exposure[node,b] = EXPMAP<node>[b] summed over src.spans[node,b]
    bkg.exposure[node,b] = EXPMAP<node>[b] summed over bkg.spans[node,b]
    D = EXPMAP<node>[b] * CanonicalCrossPsf::probability(beta(b),source)
    src.crossPSF[node,b] = D summed over src.spans[node,b]
    bkg.crossPSF[node,b] = D summed over bkg.spans[node,b]

// compute the response matrix

FOREACH incident energy bin

  // compute the loss factors specific to the narrow-featured LSF

  loss = 0
  FOREACH node
    FOREACH b = dispersion channel of node
      y = src.crossPSF[node,b]
      IF *bkgcorrect AND bkg.exposure[node,b]
        y -= bkg.crossPSF[node,b] * src.exposure[node,b] / bkg.exposure[node,b]
      loss[b] += y * ccdQuantumDataServer(node)::efficiency(energy)

  // combine with the narrow-featured LSF for each reflection order

  LSF = 0
  FOREACH order in [0,5]
    LSF += NarrowLSF(source,order,energy)
```

```
response = LSF * loss

// compute the loss factors specific to the broad-featured LSF

loss = 0
FOREACH node
  FOREACH b = dispersion channel of node
    y = empiricalCrossLA(src.spans[node,b],source,energy)
    IF *bkgcorrect
      y -= empiricalCrossLA(bkg.spans[node,b],source,energy)
    loss[b] += y * src.exposure[node,b] * ccdQuantumDataServer(node)::efficiency(energy)

// combine with the broad-featured LSF for each reflection order

LSF = 0
FOREACH order in [0,5]
  LSF += BroadLSF(source,order,energy)
response += LSF * loss

// apply loss factors that are not specific to the type of LSF;
// the redistribution function varies slowly with incident energy
// and so it is recomputed no more often than every .02 keV.
// Since version 1.13.4, the redistribution is calculated for all
// input energies.

IF redistLoss is stale
  FOREACH b = response channel
    redistLoss[b] = CanonicalRedist::spectrum(energy) summed over banana[b]
response *= redistLoss * GratingDataServer::selfVignettingEfficiency

// copy all response channels above LO_THRES to the output matrix
```

9 Comments

References