evproject

June 2, 2019

Abstract

The task **evproject** calculates linearized sky coordinates (i.e. a tangential projection on the sky) on an event-by-event basis, and stores these new coordinates in the input event list in columns `X` and `Y`, as offsets from a particular reference point.

1 Instruments/Modes

Instrument	Mode
EPIC MOS	IMAGING
EPIC MOS	TIMING
EPIC PN	IMAGING
EPIC PN	TIMING
EPIC PN	BURST

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

3.1 Genesis of evproject

This task has been developed from, and is intended to replace, the sas task **attcalc**. Why write a whole new package, rather than modifying the existing **attcalc**? This breaks into two separate queries:

1. *Q: what in* **attcalc** *needed improving?* A: it seems now to be generally accepted among the SAS development team that the coordinate transform routine employed by **attcalc** contains an error (it results in an inversion of the sign of the boresight roll angle).
2. *Q: why retain the old* **attcalc***?* A: it is mostly a question of inertia. This error has now been thoroughly built into the system and quite a lot of tasks would need to be re-written if **attcalc** were to be simply replaced by **evproject**. In fact no ill effects are felt in the

present sas, firstly because **attcalc** has itself been used to deduce the boresight angles, which are therefore themselves incorrect in this matter of roll-angle sign; secondly because other tasks which need to perform the same transform have 'cut-and-pasted' the same (incorrect!) code from **attcalc**. The errors thus cancel in the existing sas. Problems begin only for a developer who wishes to create a new task which is to calculate the same transform, and who wishes not to go on copying the same error in perpetuity.

The differences between **attcalc** and **evproject**, which are mostly structural rather than functional, are as follows:

1. The error in the coordinate transform has been corrected;
2. The transform routine is now available as a library function;
3. The construction of the attitude time series has been separated off into the library **binned_att**.

Finally, why the change of name? Because I felt that '**attcalc**' is not very descriptive of the main function of the task, which is to calculate the position of each event on a sky projection plane. When it was decided to create a new package, I thus seized the opportunity to change the name of the main task to something more suggestive of its function.

3.2 Introduction

The task **evproject** does three things:

1. For each event in the input list, **evproject** calculates an X and Y coordinate pair. These are the coordinates of the event on a plane which is tangent to the celestial sphere. The X axis of the tangent plane points in the direction of decreasing right ascension and the Y axis points in the direction of increasing declination. See section 3.3 or parameter `tangdirstyle` for a description of how the user can provide the direction of the tangent point to the program.

   These X and Y coordinate values are stored as 4-byte integers in columns of the event list designated by the parameters `xcol` and `ycol`. The coordinates are both offset and scaled, the relevant information being stored according to the usual World Coordinate System conventions in WCS column attributes of the `xcol` and `ycol` columns. The scaling is such that an increment of 1 in either X or Y value corresponds to a tangent-plane distance equal to the arc length of 0.05 arcsec (same as for the `DETX` and `DETY` columns of the event list).

   Note that the basic purpose of this projection is to make it easy to extract several images from different selections of the same events. Since the positions of all events on a projection plane are pre-calculated by **evproject**, it is a simple matter to rebin these positions to form an image. In formal terms, the X and Y values already constitute an image, albeit with a very small pixel size.

2. The task calculates a nominally average spacecraft attitude (comprising right ascension and declination of the spacecraft -X axis and the position angle of the spacecraft -Z axis) and stores this in the event list header in keywords `RA_PNT`, `DEC_PNT` and `PA_PNT`.

3. In principle, due to residual instability of the spacecraft pointing, each event is associated with a slightly different spacecraft attitude. However, because the differences from one event to another are usually slight, and because it is time consuming to perform the coordinate-transform calculation with a different attitude for each event, it was found convenient to divide the duration of the exposure into N unequal bins, each of these time bins being

associated with a fixed attitude. The task generates these bins by following the attitude and starting a new bin at the point where the attitude diverges from the bin baseline value by more than a set threshold.

Note that the task can either perform this attitude binning itself (with the option to store the result in `outbinnedattset`) or it can make use of a binning scheme already prepared by task **attbin**. See section 3.4 for details.

These activities are described in more detail in the remaining subsections of section 3.

3.3 Projecting the events

The projection direction (ie, the direction of the point at which the projection plane is tangent to the celestial sphere) is user-selectable from the following alternatives, chosen via the parameter `tangdirstyle`:

- `tangdirstyle`=nom: task uses the current nominal pointing direction of the s/c, as defined in the `RA_NOM` etc keywords of the event list;
- `tangdirstyle`=obj: task uses the pointing direction of the celestial object, as defined in the `RA_OBJ` etc keywords of the event list
- `tangdirstyle`=pnt: task uses whatever value was calculated for the `RA_PNT` etc keywords (see section 3.5);
- `tangdirstyle`=user: the task reads the RA and dec supplied by the user via parameters `tangdirra` and `tangdirdec`.

As said in the introduction, the 'pixel size' of the X and Y values is fixed at 0.05 arcsec; the remaining thing to be specified is the maximum allowed X and Y values. This is specified in a somewhat roundabout way via the parameter `imagesize`, which specifies the total half-width and/or half-height of the projection plane in decimal degrees. After offsetting by `imagesize`, X and Y can therefore run from 1 to 2$\times$ `imagesize` $\times 3600/0.05$.

3.4 Source of the spacecraft attitude information

There are three allowed sources of this information, governed by the parameter `attsource`:

1. `attsource`='binned': the binned attitude output of **attbin** is read from the file `inbinnedattset`.
2. `attsource`='odf': the attitude information stored in the ODF is used. (Note that the user must set the environment variable SAS_ODF to point to this ODF before running **evproject** in this mode.) If parameter `odfattsource` is set to 'ahf', the Attititude History File of the ODF is the source of the attitude values; if it is set to 'om', the values are read from the OM tracking history file.

   The task bins up these attitude values, as described in section 3.6; the binned-up attitude values may be exported to a file via parameters `writebinnedatt` and `outbinnedattset`. Tasks **eootemap**, **epnoisemap**, **eimchip2sky** and **eexpchipmap** are designed to be able to read this binned-attitude dataset, so that they can use the same binning scheme as **evproject**.
3. `attsource`='fixed': the attitude is taken from user-supplied values via parameters `attra`, `attdec` and `attapos`.

Note that the source chosen for attitude data affects the calculation of the PNT keywords (see next section).

3.5 Calculating the PNT keywords

The `RA_PNT`, `DEC_PNT` and `PA_PNT` keywords, together with the `AVRG_PNT` keyword which records the type of average used, are meant to describe the average value of the spacecraft attitude during the exposure. The way in which they are calculated however is controlled by the source chosen for the attitude information, as follows:

- `attsource`='binned': whatever values of the PNT keywords calculated by **attbin** and stored in the `inbinnedattset` dataset are simply copied over.
- `attsource`='odf': PNT values are sought from the header of the dataset pointed to by `pntkwdset`. This should normally be the output file of **atthkgen**, which is the SSC product file with the name string ATTTSR. The sought keywords have the form 'tsa', where 't' is either 'M' for median or 'A' for mean, 's' is either 'AHF' or 'OM', and 'a' is either 'RA', 'DEC' or 'PA'. The exact values of 't' and 's' looked for depend on the parameters `withmedianpnt` and `odfattsource`.
- `attsource`='fixed': the values `attra`, `attdec` and `attapos` are also used for the PNT output. In this case `AVRG_PNT` is set to 'FIXED'.

3.6 Binning up the attitude

When **evproject** is pointed to the ODF as the source for information on the changes in spacecraft attitude over the exposure duration (`attsource`='odf'), it automatically bins up the attitude wander in order to shorten the processing time. Each bin is associated with a start time (which is equal to the end time of the previous bin, if there is one), and end time (which is equal to the start time of the following bin, if there is one) and an attitude. The binning is done as follows. For each event time, **evproject** attempts to obtain the s/c attitude via an **oal** call. This call is either successful or unsuccessful. **evproject** begins a new attitude bin if any of the following occur:

- if the **oal** call is being made for the first time;
- if the present call was successful but the preceding call was unsuccessful;
- if the present call was unsuccessful but the preceding call was successful;
- if the call was successful, but the returned attitude varies by more than a set amount from the 'baseline attitude'.

It can thus be seen that each bin duration spans **oal** calls which were either all successful or all unsuccessful. The attitude of each new 'successful' bin is set to the baseline attitude. The baseline attitude is initially set to the attitude returned by the first succesful **oal** call. If the attitude returned by any subsequent successful **oal** call diverges from the baseline by more than set limits (item 4 in the list above), the baseline attitude is altered. However usually only one of the components of the baseline attitude is altered at a time. This comes about as follows. There are separate limits on each of the three attitude components (RA, dec and position angle). If any component of the momentary attitude diverges from the same component of the baseline attitude by more than the respective limit, that component of the baseline attitude is set to the momentary value; components which have not wandered out of bounds are left unaltered. Note also that a change of bin due to change of success of **oal** call does not in itself alter the baseline. If a new bin starts because the **oal** calls have returned to the successful state, the attitude assigned to the new bin is the baseline value, which is also the attitude of the last 'successful' bin.

This binning scheme is perhaps a little complicated, but has been chosen so as to adhere as closely as possible to the original **attcalc** scheme. However note that (i) **evpproject** does not quite adhere to the same scheme, and (ii) the component limits are calculated in a different way to **attcalc** (see below).

The limits on attitude wander are defined via the parameter `maxdelta`, which is in arcseconds. The RA limit $\Delta\alpha$ is set to

$$\Delta\alpha = maxdelta/\cos(\delta),$$

where δ is the `DEC_PNT` value. The declination limit is just set to `maxdelta`, and the position angle (apos) limit Δp is set to

$$\Delta p = maxdelta \times 60 \times 180/\pi/R_{\text{arcmin}},$$

where $R_{\text{arcmin}} = 15.0$ is the nominal radius of the field of view of the XMM EPIC cameras.

4 Parameters

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

Parameter	Mand	Type	Default	Constraints

eventset	yes	dataset		

Name of the event-list dataset.

xcol	no	string	X	

Name of the column of the event list to which to write the X coordinate values.

ycol	no	string	Y	

Name of the column of the event list to which to write the Y coordinate values.

imagesize	no	real	0.36	

Half-size of final image (in degrees).

attsource	no	string	binned	binned—odf—fixed

Source of the attitude data. If this equals 'odf', the environment variable SAS_ODF must be set to point to the appropriate ODF.

inbinnedattset	yes	dataset		

Name of the dataset from which to read the pre-binned attitude data (usually expected to be the output file of **attbin**). This parameter is read if `attsource`='binned'.

odfattsource	no	string	ahf	ahf—om

Whether to use the Attitude History File or the OM pointing history file from the ODF. This parameter is read if `attsource`='odf'.

pntkwdset	yes	dataset		

Name of the dataset from which to read the spacecraft average pointing keywords (usually expected to be the output file of **atthkgen**). This parameter is read if `attsource`='odf'.

maxdelta	no	real	0.02	0 <`maxdelta`

A new attitude bin is started if the attitude jumps by more than this amount (in arcsec). This parameter is read if `attsource`='odf'.

withmedianpnt	no	boolean	yes	

Whether to use/calculate median or mean pointing for the *_PNT keywords. This parameter is read if `attsource`='odf'.

writebinnedatt	no	boolean	no	

Whether to write the binned attitude to file. If 'yes', the data is written to `outbinnedattset`. This parameter is read if `attsource`='odf'.

outbinnedattset	yes	dataset		

The dataset which is to contain the binned attitude data. This parameter is read if `writebinnedatt`='yes'.

attra	yes	angle		$0 \leq$`attra`≤ 360

Right Ascension of the spacecraft attitude. This parameter is read if `attsource`='fixed'.

attdec	yes	angle		$-90 \leq$`attdec`≤ 90

Declination of the spacecraft attitude. This parameter is read if `attsource`='fixed'.

attapos	yes	angle		$0 \leq$`attapos`≤ 360

Position angle of the spacecraft attitude. This parameter is read if `attsource`='fixed'.

tangdirstyle	no	string	pnt	nom—obj—pnt—user

Source of celestial coordinates of the point at which the projection plane is tangent to the celestial sphere. Values 'nom', 'obj' or 'pnt' cause the task to read these coordinates from keywords in the event list which have the form *_NOM, *_OBJ and *_PNT respectively, where * is either RA or DEC. (Note that the PNT values have already been written to the file by **evproject**.) If `tangdirstyle`='user', the coordinates are read from parameters `tangdirra` and `tangdirdec`.

tangdirra	yes	real		$0 \leq$`tangdirra`≤ 360

RA of the point at which the projection plane is tangent to the celestial sphere. This parameter is read if `tangdirstyle`='user'.

tangdirdec	yes	real		$-90 \leq$`tangdirdec`≤ 90

Declination of the point at which the projection plane is tangent to the celestial sphere. This parameter is read if `tangdirstyle`='user'.

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.

badAttitudeSource *(error)*
Value of `attsource` not recognized.

badDatamode *(error)*
The `DATAMODE` keyword in the event list header must be one of ”IMAGING”, ”TIMING” or ”BURST”.

badEventTimeOrder *(error)*
The values of the `TIME` column of the event list must occur in increasing order. (Note that if this is not the case, you can sort them using task **etimedither**).

badInstrument *(error)*
The `INSTRUME` keyword in the event list header must be one of ”EMOS1”, ”EMOS2” or ”EPN”.

badOdfAttitudeSource *(error)*
Value of `odfattsource` not recognized.

badTangentPoint *(error)*
Value of `tangdirstyle` not recognized.

noDetxColumn *(error)*
Input file incorrect - no `DETX` column

noDetyColumn *(error)*
Input file incorrect - no `DETY` column

noEventsExtension *(error)*
Input file incorrect - no `EVENTS` extension

noGoodAttitudes *(error)*

noTimeColumn *(error)*
Input file incorrect - no `TIME` column

badXYNull *(warning)*
No null value is defined for X/Y columns
corrective action: Task just continues

noEvents *(warning)*
No events were found in the `EVENTS` table.
corrective action: Task finishes

outOfImageRange *(warning)*
Some events were found which have `X` or `Y` values which are outside the set ‘legal’ bounds as specified in the `TLMIN` or `TLMAX` for the appropriate column (and ultimately via the `imagesize` parameter).
corrective action: These events are ignored

6 Input Files

1. (Mandatory) Input MOS or pn event list containing correct DETX and DETY coordinates (0.05 arcsecond resolution). Keywords DATAMODE and TCDLT are read, as are (depending on tangdirstyle) RA_NOM and DEC_NOM or RA_OBJ and DEC_OBJ.
2. (Only needed if attsource='binned') **attbin** output file, as described in the documentation for the **binned_att** library.
3. (Only needed if attsource='odf') **atthkgen** output file, containing median and mean pointing direction over the observation (this is all that **evproject** accesses from this file). Keywords read are (depending on parameters odfattsource and withmedianpnt) MAHFRA, MAHFDEC and MAHFPA or MOMRA, MOMDEC and MOMPA or AAHFRA, AAHFDEC and AAHFPA or AOMRA, AOMDEC and AOMPA.

7 Output Files

1. X and Y values are written to 4-byte-integer columns in the input event list which have names given by parameters xcol and ycol. **evproject** also writes RA_PNT, DEC_PNT and PA_PNT keywords, together with the AVRG_PNT keywords.
2. (Only written if attsource='odf' and writebinnedatt='yes') a file containing the binned attitude, in the format described in the documentation for the **binned_att** library.

8 Algorithm

```
subroutine evproject

  * get input events
  * get instrument, mode and ccdid from events file
  * add X/Y columns to event table if they don't already exist

  * obtain CCF, set state

  * get refpointlabel (source of reference point information)
  * get RA\_PNT, DEC\_PNT and PA\_PNT from atthkset (atthkgen output file)
      if no file, use NOM values
  * set central reference point (nominalra, nominaldec)
  * get attitudelabel (source of attitude information)
  * if attitudelabel=fixed, get fixedra, fixeddec, fixedposangle
  * if attitudelabel=ahf or om, set attitudeFromAhf (AHF or OM)

  * loop through events
      if event time has changed (i.e. is frame different) then
        if (attitudelabel .ne. fixed) then
          call OAL_getAttitude  (AHF/OM attitude information)
          identify if special case (e.g. dec=0, +90, -90)
          transform from spherical to parallel coords
          repeat above for a point 1 degree more north along the local meridian
            (the 'help point'), to get the angle with the nominal meridian
        endif
```

```
          call getBoresightMatrix and apply boresight correction
        endif
        apply corrected attitude information to event DETX/DETY to get X/Y
    * end loop through events

    * add WCS parameters to events file
    * (calculation of and) adding of keywords to event file
    * add history to events file

end subroutine evproject
```

9 Comments

- The 'position angle' used is in actuality an '*astronomical* position angle' (see XMM-MOC-TN-0109-OAD and INT-SYS-FD-TN-0004-OAD). This angle is counted from the celestial north in the mathematically positive sense around the axis pointing from the object towards the observer (i.e. anticlockwise from the observer's point of view, looking towards the sky). This is now in line with SciSim.

10 Future developments

The task does not at present work on SDF files.

11 Examples

References