tools

August 16, 2005

Abstract

SAS tools: a collection of simple-minded but useful tools

1 Use

pipeline processing	yes
interactive analysis	yes

2 Description

The SAS tools package consists of a suite of simple programs with limited but useful functionality. In many cases they just make functionality which is present in utility/task libraries available to the end user. The following tools are available:

2.1 clex: Command-Line-Expression-evaluator

- SYNOPSIS

```
clex <expression>
```

- DESCRIPTION
  A calculator which receives a boolean or arithmetic expression as an argument, evaluates it and prints the result on standard output. If a valid arithmetic expression is given, the numeric result is printed on stdout and the program exits with 0. If a boolean expression is given, `clex` is mute and conveys the result of the evaluation to the caller via the return value: 0/1: expression evaluates to true/false.
- REFERENCE
  See **evselect** documentation for the full list of available arithmetic/boolean operators and functions.

2.2 ttag: Time-TAG-converter

- SYNOPSIS

```
ttag <tag>|<timestamp>
```

- DESCRIPTION
  `ttag` takes one argument on the command line with can be either an event list time tag as a single floating point value or a date/time specification in the form yy-mm-ddThh:mm:ss and converts it into the respective other form:

  ```
  ttag <tag>       -> yy-mm-ddThh:mm:ss
  ttag <timestamp> -> tag
  ```

  The result is printed on standard output - return value is 0 upon successful exit, > 0 otherwise.
- REFERENCE

2.3 odfexpand: ODF-FILE name expander

- SYNOPSIS

  ```
  odfexpand odfrequest=<odffilespec>
  ```

- DESCRIPTION
  `odfexpand` accepts one string argument on the command line which must be either the name of a valid ODF file [1] or an abbreviated specification following the syntax rules detailed in the descriptions of the **oal** function `OAL_selectFile`. This string is then attempted to be expand into a full ODF file name including the preceding directory name. For this to work, the environment variable

  ```
  SAS_ODF
  ```

  must point to a directory containing an ODF summary file or the directory name must be given via the generic `-o` or `--odfdir` task command line parameters.

  The result is printed on standard output - return value is 0 upon successful exit, > 0 otherwise.
- REFERENCE

2.4 lsodf: LiSt contents of ODF

- SYNOPSIS

  ```
  lsodf
  ```

- DESCRIPTION
  `lsodf` simply lists the contents of the ODF which is either pointed at by the environment variable

  ```
  SAS_ODF
  ```

  or named via the generic command line parameters `-o` or `--odfdir` (see **taskmain** for details). The specification of an ODF directory on the command line has a higher precedence than any environment setting. The result of `lsodf` is printed on standard output.
- REFERENCE

2.5 orbitdump: DUMP contents of ODF ORBIT file to stdout

- SYNOPSIS

```
orbitdump [tstep=<value>] [offsett=true|false] >orbit.dat
```

- DESCRIPTION
The tasks dumps the contents of the orbit file in the current ODF (pointed at via environment variable `SAS_ODF`) to stdout in the format `t x y z d vx vy vz vabs` where

`t`	time in Mission-Elapsed-Time [s]
`x, y, z`	components of S/C position vector in Earth-centered J2000 reference frame [km]
`d`	norm of vector (x, y, z) [km]
`vx, vy, vz`	components of S/C velocity vector in Earth-centered J2000 reference frame [km/s]
`vabs`	norm of vector (vx, vy, vz)

The time increment between successive elements is given via task parameter `tstep` which has a default value of 10s. `offsett=true` causes all time tags to be offset by the time of first element which, hence, is assigned $t = 0$.

2.6 whatatt: WHAT ATTitude at a given epoch

- SYNOPSIS

```
whatatt t=yyyy-dd-tt[Thh:mm:ss[.fff]] bstoolsout=true|false
```

- DESCRIPTION
`whatatt` computes the spacecraft attitude at a user-specified epoch t based on the contents of the attitude history file in a given ODF. The ODF to use must either be pointed at by the environment variable

```
SAS_ODF
```

or named via the generic command line parameters `-o` or `--odfdir` (see **taskmain** for details). The specification of an ODF directory on the command line has a higher precedence than any environment setting. The result of `whatatt` is printed on standard output as a tuple right-ascension/declination/astronomical-position-angle if `bstoolsout`=false. Otherwise a command line for `insbs` is generated.

- REFERENCE

2.7 satvelsrc: SATellite VELocity in SouRCe direction

- SYNOPSIS

```
satvelsrc withsrccoord=yes|no srcra=<ra> srcdec=<dec> withorthosrc=yes|no
             tdelt=<tdelt> withmjdoffset=yes|no mjdoffset=<intmjd>
             withexplicittimes=yes|no tfrom=<t0> tto=<t1>
```

- DESCRIPTION
  `satvelsrc` computes as a function of time the component of the spacecraft velocity vector in the direction of given celestial source. The source coordinates are obtained from the summary file in the ODF being pointed at by the environment variable

  ```
  SAS_ODF
  ```

  or named via the generic command line parameters `-o` or `--odfdir` (see **taskmain** for details). The specification of an ODF directory on the command line has a higher precedence than any environment setting. The source coordinates obtained from the ODF can be overridden with the command line paramters `withsrccoord=yes srcra=<ra> srcdec=<dec>` and `withorthosrc=yes` defines a fictitious source in the direction perpendicular to the orbital plane. In this case the velocity component should be constantly zero at all times which is an important check for the integrity of the orbit file in the ODF.

  The velocity is printed as a function of time (units: days - integer part of MJD) in units of km/s in steps of `tdelt` to `stdout`. The time range is automatically chosen to be the duration of the observation but can also be explicitly given via the command line parameters `withexplicittimes=yes tfrom=<t0> tto=<t1>`. The parameters `withmjdoffset=yes mjdoffset=<intmjd>` define an integer MJD number by which all times will be offset with - if omitted the values shall be computed from the start of the observation.

- REFERENCE

2.8 tcsfix: Time Correlation Spacecraft file FIXing task

- SYNOPSIS

  ```
  tcsfix [clonetcs=yes|no] [rerun=yes|no]
  ```

- DESCRIPTION
  This task fixes a number of distinct problems in ODF time correlation data:

  1. Due to wrong ground station and spacecraft time propagation delays hardcoded in XSCS<=v71 all time correlation files produced with these XSCS versions contains too small UTC values. The absolute error is - depending slightly on the GS ID - of the order of 60ms. In addition some of the early TCS files do not contain the ERTIMRAW column that hold the non-propagation delay corrected earth reception times. Also, some early TCS have bogus entries with UTCDAY=0 - those rows are removed from the table.
  2. During revolutions $384-394$ (both inclusive) the Santiago ground station clock was running exactly one second ahead of UTC. Consequently, the time correlation data show a corresponding discontinuity at the time of a station handover to Santiago (GSID==41).
  3. During several periods in 2002 timing data from the Kourou ground station showed various anomalies which manifests themselves as discontinuities at the time of a handover to/from this station.

  `tcsfix` does:

  - remove rows where UTCDAY=0
  - add the ERTTIMRAW column to the TCS unless it exists already

  C1 if (rev ¡ 207)

  * undo the wrong propagation delays to all UTC fields and applies the correct delays

C2 if (rev >= 384 && rev <= 394)

* subtract 1s to UTC data from Santiago station

C3 if (observation start/end time lies in a Kourou bad-time window)

* correct anomalies

– adds attribues

```
TCSFIXED = T // this TCS has been fixed
TCSFIXV = M.m.p // tcsfix version number
TCSFIXC = id // numerical code of applied corrections
```

C1 is only done if the revolution number is less than 207 (XSCS-v72 was put online at the beginning of rev 207); C2 is only done for ODFs from revolutions [384, 394]; C3 is only done if observation start/end time lies within affected time windows. C1/C2/C3 are not performed if the attribute `TCSFIXC` indicates that these corrections have been applied already in a previous run (to avoid double-correction through repeated invocations of `tcsfix`. Each set bit in `TCSFIXC` stands for one distinct applied correction, i.e. `TCSFIXC=0` indicates that the data has not been modified.

2.9 xmmtrack: XMM orbit TRACKing

- SYNOPSIS

```
xmmtrack
```

- DESCRIPTION

`xmmtrack` is an interactive IDL application that visualizes the orbit and attitude data in an ODF in the form of a 3-D animation sequence. The presented scenery includes an astronomically accurate simulation of the Earth-Sun system plus the actual and predicted trajectory of the XMM satellite on its path around the Earth. In more detail, the following graphical elements are present:

– Earth:
The central body in the view frame which is illuminated by the Sun on its day side. The Earth's surface shows the main geological features (continents, oceans) and is rotating eastward at a constant rate of 1 per day. The center defines the origin of a Cartesian reference coordinate system whose three orthogonal axes are shown as red, green, and blue lines. They define the directions: $(RA = 0/Dec = 0), (RA = 90/Dec = 0), (Dec = 90)$ respectively.

– Actual trajectory:
A thin yellow three dimensional line outlines the actual trajectory of the satellite that this ODF (pointed at via the environment variable `SAS_ODF`) corresponds to. The end points mark the beginning and the end of the observation.

– Predicted trajectory:
A thin red lines marks a complete orbit around the Earth for a specified epoch (see below). Please note: XMM's orbit is highly time variable and the predicted orbit will significantly deviate from the actual trajectory if the reference epochs differ. The predicted orbit is calculated on the bases of interpolated TLE (two-line-element) data from NASA/GSFC's Orbital Information Group.

– Satellite:
The satellite is shown as a solid model with a main cylindrical body and rectangular solar panels in its actual attitude. A thin red line signifies the viewing direction, the blue line is the $+Z$ axis and the green line completes the orthogonal right-handed

triad. Please note: Under normal circumstances one side of the solar panels is fully illuminated by the Sun at all times. If there is no attitude data available the satellites attitude will be set to RA=Dec=APOS=0 as a result of which the solar panels can become unilluminated.

- GUI CONTROLS:
  The user can interactively change the displayed scenery with a couple of GUI elements:
  - Drawing area:
    The drawing area is sensitive to left-button mouse clicks. Moving the mouse while holding the left button pressed allows to rotate the scenery arbitrarily about all three axes.
  - Animation button group:
    Three buttons labeled *Start*, *Stop*, and *Reset* to start, stop and reset the animation sequence.
  - Satellite Scaling:
    A slider to change the size of the satellite in the displayed frame between 100% and 900%
  - Zoom view:
    A slider to zoom into the view - range is $[1, 10]$ with 1 being the farthest and 10 providing the closest view of the scenery.
  - Animation step:
    A slider to tune the animation step size - range: [100, 5000]s.
  - Time slider:
    The time slider at the bottom allows to advance the scenery to a specific epoch. If the animation is running, push the *Stop* button, and move the slider to the desired position. To resume the animation from this new point onward simply push the *Start* button.
  - Predicted orbit:
    By default the predicted orbit is not shown in the scenery. Pushing the button labeled *Add* will display a full predicted orbit for the displayed reference epoch. The latter is changeable, i.e., the time dependence of the orbit evolution can easily be studied. The predicted orbit is shown with two white reference lines to guide the eye. The short one signifies the position of the perigee. The longer one connects the ascending and descending nodes of the orbit. Those are the points where the orbital plane intersects the equator. Together with the orbit the three main orbital elements are shown in the left panel:
    * orbit inclination
    * R.A. of the ascending node
    * argument of the perigee
- USED SAS COMPONENTS
  - **oal**: inquire orbit/attitude data
  - **caloalutils**: compute predicted orbit
  - *utils*: various STime routines

2.10 regionmask: convert FITS REGION to image MASK

- SYNOPSIS

```
regionmask region=<table> autosize=no withmaskset=yes maskset=mask.ds
           whichpixdef=image pixdefset=<image>
           whichpixdef=columns xcolumn=<column> ycolumn=<column>
           whichpixdef=manual xrpix=1 xrval=1 xdelt=1 xbins=100
                              yrpix=1 yrval=1 ydelt=1 ybins=100
```

- DESCRIPTION

`regionmask` converts a FITS Region extension table to an image mask, either as a FITS image or as an ASCII list of pixels. The FITS Region extension format describes an arbitrary selection region in terms of set operations on basic geometrical shapes. This format has some virtues to recommend it (most particularly its compactness), and is becoming a standard representation, but at present it remains awkward and poorly supported for most users. Once converted to an image mask the region can be viewed without specialized software and can be used for image processing. An image mask is a raster image (array of pixels) with all pixels outside the region set to zero and all pixels inside the region set to one. This task uses the `ShapeScanner` module of **rgslib** to rasterize the region.

The mapping from the region coordinate system to image coordinates is established via the command line parameters: a template may be provided in the form of an existing FITS image or pair of FITS table columns, or the user may specify the mapping explicitly. These same parameters also define a clipping rectangle (the origin and extent of the output image), which engages by default. An alternative clipping rectangle, the "natural" bounding box of the region itself, may be substituted by setting parameter `autosize`=*yes*. The natural bounding box of a region is the smallest rectangle outside of which all pixels have identical value. However, some regions (sectors, for example) have no natural bounding box, and auto-sizing these will yield an output image of "infinite" size, which will not be aborted gracefully by `regionmask`. Note also that for a complex region the auto-sized clipping rectangle computed by `regionmask` is generally larger than its true natural bounding box. These problems do not seem to warrant solutions at this time.

Parameter `whichpixdef` activates one of the following specifications for the mapping from region coordinates to image coordinates:

- *image*

  This is the default. Parameter `pixdefset` is the name of a FITS image (PRIMARY array) with World Coordinate System (WCS) keywords defined on the first and second axes. The output image will use the same pixel size and alignment, and (unless auto-sized) the same origin and extent.

- *columns*

  Parameters `xcolumn` and `ycolumn` are the names of FITS table columns with WCS and bounding keywords. The preferred bounding keywords are TLMIN and TLMAX, but in their absence TDMIN and TDMAX will be used instead. The WCS keywords specify the size and alignment of the pixels and the bounding keywords specify the clipping rectangle of the output image (unless auto-sized). Its first and second axes are specified by `xcolumn` and `ycolumn` respectively.

- *manual*

  Parameters `xrpix`, `xrval`, `xdelt`, and `xbins` explicitly define the first axis of the output image. Parameters `yrpix`, `yrval`, `ydelt`, and `ybins` explicitly define its second axis. (rpix,rval,delt) is a WCS mapping: delt is the grid spacing, rval is a reference point in the region coordinate system, and rpix is the image coordinate associated with that reference point. Unless auto-sized, bins is the extent of the output image, and its origin is placed at the reference point.

The default output format (parameter `withmaskset`=*yes*) is a FITS image (PRIMARY array) of unsigned bytes. Its WCS keywords record the mapping from image coordinates back to the region coordinate system. Parameter `maskset` is the file name.

The alternate output format (parameter `withmaskset`=*no*) is sent to the tty standard out in three columns of ASCII text: pixel x-coordinate, low pixel y-coordinate, and high pixel y-coordinate. The latter two columns define a vertical span of pixels that are all inside the region. Multiple spans at the same x-coordinate are output on successive lines, such that the output is sorted in increasing order of pixel coordinates (x first, then y). The output is restricted to the clipping rectangle, and x-coordinates within that rectangle are skipped unless associated with at least one y-coordinate span. Note that these coordinates refer to the reference pixel of the given region-to-image coordinate mapping—not the lower-left corner of the clipping rectangle.

3 Comments

none.

4 Future developments

add more tools

References

[1] ESA. XMM Interface Control Document: Observation and Slew Data Files (XSCS to SSC) (SciSIM to SOCSIM). Technical Report XMM-SOC-ICD-0004-SSD Issue 2.5, ESA/SSD, June 2000. Found at the URL: `ftp://astro.estec.esa.nl/pub/XMM/documents/odf_icd.ps.gz`.