SWAN Analysis Software: Version 5

Eric Quémerais, Stéphane Ferron

Updated: September 2005

Abstract

This is a presentation of various softwares which have been developed at Service d'Aéronomie to allow for a quick analysis of the SWAN data. They mainly deal with computation of the direction of observation, calibration and data binning to create images.

All these programs and utilities are based upon the **IDL** software and should be compatible with the different UNIX machines or PC's equipped with versions 5.0 and higher of **IDL**.

The first section presents the different files containing these programs and how they should be used. Then, the various local variables are defined.

The structures of the programs in routines, functions and main program are underlined as well. Finally, we present all the subroutines, functions and their default parameters.

1 Loading Data and Computing an Image

Name of Progam: **SwanImageTool.pro**

This program loads a SWAN data set stored under the FITS format.

The main variables are described at the following link:
Donner le nom du lien

The list of tasks performed by the program is the following:

- The program loads and converts the Swan Observation data stored in the FITS file.
- The program loads the SOHO orbit datafile and the motor coordinate correction files.

- The program applies motor coordinate corrections to the data to account for measured offsets.
- The program computes a flatfield correction to compensate for Pixel Response Non-Uniformity.
- If data from both sensors are present (full-sky images), the programme computes the +Z/-Z cross calibration factor. Data of the -Z sensor are converted to equivalent +Z sensor count rate. Throughout the mission the +Z sensor has been used as a reference for the instrument calibration.
- The program computes an image in a chosen coordinate system. The size and step size of the image can be chosen.

The output of the program is a set of images. If data from both sensors are present, there will be an image for the +Z sensor and an image for the -Z sensor.

The results are stored in a 3-dimension variable like that:

```
COUPOFF = FLTARR( N1, N2, 2)
```

The first 2 dimensions N1 et N2 correspond to the size of the image.

It is defined for N1-1 longitude values and N2 latitude values. The `N1-1` line is used to define the limits of the image.

The array given by `COUPOFF(0:N1-2,0:N2-1,0)` contains the image obtained from the data.

The array `COUPOFF(0:N1-2,0:N2-1,1)` contains the statistical error associated to the image.

Spatial coordinates of the point `(i,j)` are given by:

```
longitude(i,j) = lonmin + (i+0.5) × londel
 latitude(i,j) = latmin + (j+0.5) × latdel
```

The limits of the image are given by

```
  lonmin = COUPOFF(N1-1,0,1)
  londel = COUPOFF(N1-1,1,1)
  latmin = COUPOFF(N1-1,2,1)
  latdel = COUPOFF(N1-1,3,1)
lonmax = lonmin + (N1-1) × londel
  latmax = latmin + N2 × latdel
```

2 SOHO Orbit file

The following file contains the information about the position of SOHO

orbitsohoidl.dat

The variables are defined in heliocentric ecliptic (J2000) coordinates

- `TEMPSOHO`: Time in days since 01/01/1995
- **SOHO position vector**
- `RR_SOHO`: SOHO heliocentric distance in km.
- `XX_SOHO`: 1^{st} normalized component.
- `YY_SOHO`: 2^{nd} normalized component.
- `ZZ_SOHO`: 3^{rd} normalized component.
- **SOHO Velocity Vector**
- `VX_SOHO`: 1^{st} component in km/s.
- `VY_SOHO`: 2^{nd} component in km/s.
- `VZ_SOHO`: 3^{rd} component in km/s.
- **Earth position as seen from SOHO**
- `RTERRE_SOHO`: Earth-SOHO distance in km.
- `XTERRE_SOHO`: 1^{st} normalized component.
- `YTERRE_SOHO`: 2^{nd} normalized component.
- `ZTERRE_SOHO`: 3^{rd} normalized component.
- **Earth position as seen from the Sun**
- `RTERRE_SOL`: Sun-Earth distance in km.
- `XTERRE_SOL`: 1^{st} normalized component.
- `YTERRE_SOL`: 2^{nd} normalized component.
- `ZTERRE_SOL`: 3^{rd} normalized component.

There are 5 points per day in the file.
The file starts at (`TEMPSOHO`(0) = 365.0) 01/01/1996.

3 Useful Routines

- **function** `reflection`(normale,vector-in)

  Computes the direction of a light ray reflected on a mirror.

- **function** `su2soho`(outrot,inrot,xpixel,ypixel,suz)

  Computes the transformation from motor angles and pixel position into SOHO reference frame.

  The output is a 3D vector defining the line of sight.

  outrot = outer motor position in degrees
  inrot = inner motor position in degrees
  xpixel and ypixel coordinates of the pixel on detector array
  suz equals 1 for SU +Z and -1 for SU -Z.

- **function** `su2soho25`(outrot,inrot,suz)

  Same as previous. For all pixels of the 5x5 array. The routine `su2soho27` computes for the 5x5 array plus the 2 outlying pixels.

- **function** `soho2su`(visee-soho)

  Conputes the transformation from SOHO coordinates to motor positions (central pixel).

- **function** `soho2ecli`(sc-posit,coord-in,inv)

  Computes the transformation from SOHO coordinates to ecliptic coordinates (J2000).

  inv = -1 makes the inverse transformation (ecliptic to SOHO).

- **function** `ecli2equa`(coord-in,inv)

  Ecliptic to Earth Equatorial coordinates (J2000). inv = -1 makes the inverse transformation.

- **function** `equa2gala`(coord-in,inv)

  Earth Equatorial coordinates to Galactic coordinates. inv = -1 makes the inverse transformation.

- **function** `ecli2wind_oy`(coord-in,inv)

  Ecliptic to wind-Oy coordinates. inv = -1 makes the inverse transformation.

- **function** `ecli2wind`(coord-in,inv)

  Ecliptic to wind-Ox coordinates. inv = -1 makes the inverse transformation.

- **function** `convert3to2rad`,coordinate

  Conversion from cartesian to spherical coordinates (in radians).

- **function** `convert3to2deg`,coordinate

  Conversion from cartesian to spherical coordinates (in degrees).

- **function** `convert2degto3`,angles

  Conversion from spherical (degrees) to cartesian coordinates.

- **function** `convert2radto3`,angles

  Conversion from spherical (radians) to cartesian coordinates.

- **function** `add`(coef1,ima1,coef2,ima2)

  adds to images (coupoff format) using coefficients coef1 and coef2.

- **pro** `plotimage`,coupoff,minicol,maxicol (,limites=limites,slon=slon,slat=slat,vco=vco)

  2D representation of an image coupoff. Only values between minicol and maxicol are represented.

  If the array limites (lonmin,lonmax,latmin,latmax) is defined, the plot is restricted to the chosen interval. The sky projection is bosed on the map_set IDL routine.

- **pro** `plot2image`,coupof1,coupof2,minicol,maxicol (,limites=limites,coef=coef)

  2D representation of 2 images coupof1 and coupof2 on the same figure. Only values between minicol and maxicol are represented.

  if coef is defined, coef2 is multiplied by this factor.