DEPARTMENT OF SPACE RESEARCH
BIRMINGHAM UNIVERSITY
PROG/011
Asterix Data Analysis
ASTERIX Programming Note 011

DJ Allan
Revised 30 April 2000

The ASTERIX PSF System Routines

Contents

1 Introduction

This document describes the ASTERIX PSF system in order that users can write applications to use the PSF routines or write their own PSF data routines.

A description of the user interface can be found in the ASTERIX on line help system ASTHELP (type Asthelp PSF).

2 Overview

ASTERIX applications are designed to be instrument independent. The PSF System routines maintain instrument independence by providing a consistent interface. Any instrument specific code is kept in PSF library routines. The PSF library routines must use the PSF system interface to interact with the application.

```
application   <->  PSF system  <->  PSF data routines
```

3 The Application Level

3.1 Initialisation

Any application using the PSF system must perform all access between calls to the two system routines PSF_INIT and PSF_CLOSE. The former ensures that the common block is explicitly initialised (vital for monoliths) and the latter that all memory/file resources are released by the system before the application returns control to the calling environment.

Note that the INIT and CLOSE routines, unlike most such pairs in ASTERIX subpackages, require a STATUS argument. This is for the simple reason that they can generate errors when executing.

3.2 Associating a Dataset with a PSF

The PSF system associates a dataset with a PSF using one of the two routines PSF_ASSOCI, PSF_ASSOCO, depending on whether read or write access is being performed. Both routines return a PSF reference number to the application, which should be declared as type INTEGER. All subsequent access to PSF system routines should pass this reference number as the first argument. More than one PSF can be associated with the same dataset, and more than one dataset can be associated simultaneously. The total number of active references is defined by the symbolic constant PSF_NMAX in the ASTERIX include file PSF_PAR.

Before discussion of the routines which return useful data, the routine PSF_RELEASE should be mentioned. This simply frees all internal storage used by the PSF system for a specified reference.

3.3 Accessing PSF data

Discussion of the arguments to PSF routines is not complete without a definition of the coordinate system used.

```
+---------------------------------------------------+
|                                                   |
|                                                   |
|                            ^ +-----------------+  |
|                            | |                 |  |
|                            | |   +             |  |
|                            NY|(X0,Y0) +        |  |
|                            | |        (X0+QX,  |  |
|                         +  | |         Y0+QY)  |  |
|                      (0,0) v +------ARRAY------+  |
|                              <--------NX------->  |
|                                                   |
|                                                   |
|                                                   |
|                                                   |
|                                                   |
+---------------- dataset X,Y space ----------------+
```

The dataset associated with a PSF must have enough axis information to define the X,Y space in angular units. The ASTERIX standards document PROG_002 defines the pointing direction as (0,0) in this space.

The most primitive PSF access routine is PSF_2D_DATA which directly drives the PSF data routines. Other system routines such as the profilers RADIAL_PFL and ENERGY_PFL are built on top of PSF_2D_DATA. The argument list of PSF_2D_DATA is,

```
PSF_2D_DATA( SLOT, X0, Y0, QX, QY, DX, DY, INTEG, NX, NY,
                                          ARRAY, STATUS )

  SLOT <int>    The PSF reference
  X0,Y0 <real>  The image position at which the PSF is to
                  be evaluated, in radians
  QX,QY <real>  Radian offsets from the position (X0,Y0)
                  to the centre of the array ARRAY
  DX,DY <real>  The widths of the bins in ARRAY
  INTEG <log>   Return integrated probability per pixel?
  NX,NY   <int>   The dimensions of ARRAY
  ARRAY(NX,NY) <real>  The 2D probability array
```

The units of the values in ARRAY when INTEG is true are integrated probability per unit pixel area DX*DY. Thus, if the image area ARRAY is sufficiently large, the sum of the values returned should approach or even equal unity. The units of if INTEG is false are currently undefined.

The discussion above has mentioned nothing of potential time or energy dependence of the PSF. Most application software is optimised to handle spatial variation efficiently, with time and energy being treated as extensions. The PSF system is organised to reflect this method of programming - the data routines only provide control over the spatial variables. To change the energy or time separate calls to PSF_DEF must be made. This allows time and energy bands, as well as user definable data to be passed to the PSF data routine. A typical structure for processing such data where the PSF also varies with image position might look like,

```
CALL PSF_ASSOCI( LOC, SLOT, STATUS )
DO E = 1, NCHANNEL
  CALL PSF_DEF( SLOT, ..., STATUS )
  DO X = 1, NX
    DO Y = 1, NY
      CALL PSF_2D_DATA( SLOT, ... STATUS )
    END DO
  END DO
END DO
```

Note that the PSF system routines know nothing about time or energy. The sole means by which the PSF_DEF call has any effect is by altering the internal state of the PSF library routine generating the 2D PSF data.

3.4 A Data routine

For a PSF of name <name>, the data routine is called PSF_<name>. The arguments to this routine are specified in Section 5.

The data routine will usually require some ancillary information to generate PSF data. Such data usually only needs to be accessed once when the PSF is selected by the user. The means to do this is through the PSF_<name>_INIT routine.

3.5 Data _INIT routine

When a PSF is selected using one of the PSF_ASSOC routines, the routine PSF_<name>_INIT is invoked if present. This routine is supplied with the slot number and locator to the associated dataset. The former can be used to index a common block inside the library if required - the locator is available for any access the library routine might need, eg. getting a filter id.

Note that the _INIT routine should never annul the locator it is given, and should either free any resources it uses or store them in COMMON so that the PSF_<name>_CLOSE routine can do so.

The PSF system defines two ADAM parameters called MASK and AUX for the use of _INIT routines. These can be used to get auxilliary information from the user The interface file parameter type for both is "LITERAL" which means that scalar values and filesnames can be read using these parameters; vectors, however, must be parsed after reading PAR_GET0C. The prompts require defining using the PAR_PROMT subroutine. _INIT routines should always

PAR_CANCL any environment parameter accessed. This permits multiple calls to an _INIT routine for different PSFs.

3.6 Data _CLOSE routine

This routine, if present, is called by PSF_RELEASE or PSF_CLOSE when a PSF is being released from the PSF system. Its usual function is to free any resources allocated by the <name> or <name>_INIT routines.

3.7 Data _DEF routine

The PSF_<name>_DEF routine is invoked, if present, by a call to the PSF system routine PSF_DEF. This link is provided in order that any internal settings stored in COMMON by a PSF data routine and associated routines can be changed. The seetings can then be access by the data routine.

3.8 Data _PFL routine

The PSF_<name>_PFL routine provides an mechanism to generate energy profile values directly. The PSF system routine PSF_ENERGY_PFL has an algorithm to perform energy profiling by repeated calls to the routine. However, this routine is susceptible to PSFs with discontinuities and is very slow. If energy radii can be calculated analytically for your PSF it is best to write this routine.

4 The PSF system routines

The following subset of the PSF system routines are for use by application software. Note that ";" in argument lists shows the break between the input arguments to the left, and output arguments to the right.

```
PSF_INIT( STATUS )
```

Initialise the PSF common block and load the available PSF libraries.

```
PSF_CLOSE( STATUS )
```

Free all PSF system sources

```
PSF_ASSOCI( LOC; SLOT, STATUS )

 LOC <char>   Locator to dataset
  SLOT   <int>     The PSF reference
```

Associate a dataset with a PSF using READ access

```
PSF_ASSOCO( LOC; SLOT, STATUS )

 LOC <char>   Locator to dataset
  SLOT   <int>     The PSF reference
```

Associate a dataset with a PSF using WRITE access.The name of the PSF selected is written to the dataset.

```
PSF_RELEASE( SLOT, STATUS )

  SLOT   <int>     The PSF reference
```

Free any resources used by the PSF system for the PSF SLOT

```
PSF_2D_DATA( SLOT, X0, Y0, QX, QY, DX, DY, INTEG, NX, NY,
                                          ARRAY, STATUS )

  SLOT   <int>     The PSF reference
  X0,Y0  <real>    The image position at which the PSF is to
                   be evaluated, in radians
  QX,QY  <real>    Radian offsets from the position (X0,Y0) to
                   the centre of the array ARRAY
  DX,DY  <real>    The widths of the bins in ARRAY
  INTEG  <log>     Return integrated probability per pixel?
  NX,NY  <int>     The dimensions of ARRAY
  ARRAY  <real>[]  The 2D probability array
```

Return a 2D array of probability per unit pixel for the PSF specified by SLOT.

```
PSF_DEF( SLOT, LOWT, HIGHT, LOWE, HIGHE, USERIN, USEROUT,
                                                STATUS )
  SLOT   <int>     The PSF reference
```

```
  LOWT    <dble>    The lower time bound in offset from OBS_TAI
  HIGHT   <dble>    The upper time bound in offset from OBS_TAI
  LOWE    <int>     The lower energy channel bound
  HIGHE   <int>     The upper energy channel bound
  USERIN    *       User input to library DEF routine
  USEROUT   *       User output from library DEF routine
```

Define the current energy and time bands for subsequent PSF evaluation. The time units are offsets from OBS_TAI, ie. the TAI equivalent of the time origin of the dataset, and the energy units are channel numbers. Additional USERIN and USEROUT arguments are provided for other possible information which could affect the PSF.

```
PSF_ENERGY_PFL( SLOT, NFRAC, FRAC, X0, Y0, RADII, STATUS )

  SLOT    <int>     The PSF reference
  NFRAC   <int>     Number of energy fractions to be evaluated
  FRAC    <real>    Array of fractions for radii to be found
  X0,Y0   <real>    The image position at which the PSF is to
                    be evaluated, in radians
  RADII   <real>[]  The radii in radians
```

Returns the radius in radians at which the specified PSF SLOT encloses each of the NFRAC energy fractions in FRAC. The PSF is evaluated at the image position (X0,Y0).

```
PSF_QMODEL( SLOT, MODEL, STATUS )

  SLOT    <int>     The PSF reference
  MODEL   <logical> PSF is a model
```

Returns a logical flag depending on whether the PSF SLOT was defined using a PSF model specification.

```
PSF_RADIAL_PFL( SLOT, X0, Y0, DIM, PROF, STATUS )

  SLOT    <int>     The PSF reference
  X0,Y0   <real>    The image position at which the PSF is to
                    be evaluated, in radians
  DIM     <int>     Number of profile values
  PROF    <real>[]  Profile values
```

Returns an normalised surface brightness profile of the specified PSF SLOT, evaluating the PSF at (X0,Y0).

```
PSF_RESAMPLE( NX, NY, IN, BORDER, OX, OY, OUT, STATUS )
  NX,NY  <int>     Dimensions of IN and OUT arrays
  IN     <REAL>[]  The input 2D PSF
  BORDER <int>     Border in pixels
  OUT    <REAL>[]  The input 2D PSF
```

Used to shift a 2D array of values by an amount less than or equal to one pixel in each dimension. The value of BORDER which should be at least 1 prevents edge effects affecting the values in OUT within this BORDER. The units of OX and OY are fraction pixels.

5 Data Routines

The routines below are defined for a PSF of name ¡name¿. Only the PSF_¡name¿ routine is mandatory.

```
PSF_<name>( SLOT, X0, Y0, QX, QY, DX, DY, INTEG, NX, NY,
                                         ARRAY, STATUS )

  SLOT   <int>     The PSF reference
  X0,Y0  <real>    The image position at which the PSF is to
                   be evaluated, in radians
  QX,QY  <real>    Radian offsets from the position (X0,Y0) to
                   the centre of the array ARRAY
  DX,DY  <real>    The widths of the bins in ARRAY
  INTEG  <log>     Return integrated probability per pixel?
  NX,NY  <int>     The dimensions of ARRAY
  ARRAY  <real>[]  The 2D probability array
```

Return a 2D array of probability per unit pixel for the PSF specified by SLOT.

```
PSF_<name>_INIT( SLOT, LOC, STATUS )

  SLOT   <int>     The PSF reference
  LOC    <char>    Locator to associated dataset
```

```
PSF_<name>_DEF( SLOT, SLOT, LOWT, HIGHT, LOWE, HIGHE, USERIN,
                                            USEROUT, STATUS )
```

```
  SLOT   <int>     The PSF reference
  LOWT   <dble>    The lower time bound in offset from OBS_TAI
  HIGHT  <dble>    The upper time bound in offset from OBS_TAI
  LOWE   <int>     The lower energy channel bound
  HIGHE  <int>     The upper energy channel bound
  USERIN   *       User input to library DEF routine
  USEROUT  *       User output from library DEF routine
```

```
PSF_<name>_CLOSE( SLOT, STATUS )

  SLOT   <int>     The PSF reference
```

```
PSF_<name>_PFL( SLOT, NFRAC, FRAC, RADII, STATUS )

  SLOT   <int>     The PSF reference
  NFRAC  <int>     The number of radii to evaluate
  FRAC   <real>[]  The energy radii required
  RADII  <real>[]  The radii in radians for each energy fraction
```