

spi_fitlib

User Manual

Version 1.3.0
3 October 2006

Jürgen Knödseder
Centre d'Etude Spatiale des Rayonnements
knödseder@cesr.fr
<http://www.cesr.fr/~jurgen/index.html>

Note to the user

This software has been written to analyse data of the SPI telescope onboard INTEGRAL. Particular care has been taken in making the software user friendly and well documented. If you appreciated this effort, and if this software and User Manual were useful for your scientific work, the author would appreciate a corresponding acknowledgment in your published work.

Contents

1	Introduction	1
2	Getting started	1
3	spi_fitlib data structures	2
3.1	SPIFitPar	2
4	Using spi_fitlib	5
4.1	SPIFit_Init	5
4.2	SPIFit_PrepareFit	5
4.3	SPIFit_DoFit	6
4.4	SPIFit_Filter	6
4.5	SPIFit_ScaleModels	6
4.6	SPIFit_GetSkyPars	7
4.7	SPIFit_GetBackPars	7
4.8	SPIFit_GetBounds	7
4.9	SPIFit_GetTime	7
4.10	SPIFit_DumpFitStatus	7
4.11	SPIFit_DumpFitPar	7
4.12	SPIFit_DumpSkyPars	7
4.13	SPIFit_DumpBackPars	7
4.14	SPIFit_DumpTime	7
4.15	SPIFit_WriteFitResults	7
4.16	SPIFit_DumpResiduals	7
4.17	SPIFit_DumpChisq	8
4.18	SPIFit_model	8

1 Introduction

The library `spi_fitlib` comprises ANSI C++ routines that allow the fitting of SPI data-space models. The library is used by the SPI science analysis executables `spi_obs_fit` and `spi_obs_mrem`.

2 Getting started

Before installing `spi_fitlib`, make sure that the ISDC support platform 6.3 or higher is installed on your system and that the `spi_toolslib` is available.

After downloading the `spi_fitlib.tar.gz` file, step into a directory that should hold the distribution, move the `spi_fitlib.tar.gz` file into this directory and type after the UNIX prompt `$` (don't type this prompt):

```
$ gunzip spi_fitlib.tar.gz
$ tar xvf spi_fitlib.tar
```

The first command uncompresses the distribution file, the second unpacks the files.

Before configuration, the distribution needs to be reset to a clean state. To do this, type

```
$ make distclean
```

Then, configure the distribution. It is assumed here that you have previously installed the ISDC support platform, thus you should type

```
$ ${ISDC_ENV}/bin/ac_stuff/configure
```

Finally, build the distribution by typing

```
$ make global_install
```

To perform a unit test, type

```
$ make test
```

3 spi_fitlib data structures

3.1 SPIFitPar

Each data-space model component is described by a SPIFitPar data-structure. This structure has the format

```
typedef struct {
    ModelType    type;                // General model parameter definition
    char         name[MAX_CHAR];      // Model type
    char         ebounds[MAX_CHAR];   // Model name
    char         method[MAX_CHAR];    // Model energy boundaries
    int          fixed;               // Model fit method
    int          gedfail;             // 1: Model is fixed (not fitted)
    int          hasMin;              // 1: Consider GeD failing
    int          hasMax;              // 1: Model has minimum constraint
    double       initial;             // 1: Model has maximum constraint
    double       scale;               // Initial model parameter
    double       min;                 // Global model scaling factor
    double       max;                 // Model parameter minimum
    double       time;                // Model parameter maximum
    SPICoordSys  srcSystem;           // Global time parameter (sec)
    SPIPixelUnit srcUnit;             // Source coordinate system
    double       srcXPos;             // Source flux unit
    double       srcYPos;             // Source X position
    SPIImage     srcImage;            // Source Y position
    SPIBounds    srcEbounds;          // Source image to be fitted
    DSGroup      grp_pt;              // Source energy boundaries
    DSGroup      grp_det;             // Pointing group type
    DSGroup      grp_ebin;            // Detector group type
} SPIFitPar;
```

The members have the following meaning:

- **type** [MODEL_INVALID]

Describes the type of the model. The `ModelType` may be one of the following: `MODEL_IMAGE` for an image that has been convolved with the instrumental response, `MODEL_DATASPACE` for a data-space model, `MODEL_SOURCE` for a point source model, and `MODEL_INVALID` in any other case.
- **name** []

Gives the name of the model in a user-readable format. For a point source (`type = MODEL_SOURCE`) the name is automatically constructed from the source coordinates.
- **ebounds** []

Gives the energy range that is covered by the model in a user-readable format.
- **method** []

Gives the fit method that should be used to adjust the model component to the data in a user-readable format.
- **fixed** [0]

Specifies if the model component is fixed (if set to 1) to if it should be fitted (if set to 0).
- **gedfail** [0]

Specifies if Germanium detector failures should be respected for model component fitting; if set to 1, then distinct model scaling factors will be used for all periods with a specific detector configuration.

- **hasMin** [0]
Specifies if model component has a minimum parameter value constraint.
- **hasMax** [0]
Specifies if model component has a maximum parameter value constraint.
- **initial** [1.0]
Specifies the initial value for all parameters that are associated with this model component. By default, the initial value is set to 1.0.
- **scale** [1.0]
Specifies a scaling factor by which the parameters are divided internally. This allows to bring internal parameter values all to the same scale, while fitting parameters that inherently have different scales (such as point source fluxes together with diffuse emission model scaling factors). The initial internal parameter values are given by `par = initial / scale`; the parameter values that are written to the `log` or `result` files are given by `value = par * scale` (where `par` is the internally handled parameter value). By default, the scaling factor is set to 1.0.
- **min** [0.0]
Specifies the minimum parameter value constraint (if `hasMin = 1`).
- **max** [0.0]
Specifies the maximum parameter value constraint (if `hasMax = 1`).
- **time** [0.0]
Specifies the time interval.
- **srcSystem** [SPI_GALACTIC]
Specifies the coordinate system in which a point source has been specified for the model. Values of `SPI_GALACTIC` and `SPI_CELESTIAL` are allowed. Only used for point source models (`type = MODEL_SOURCE`).
- **srcUnit** [SPI_UNIT_UNKNOWN]
Specifies the unit in which the source flux is given. This parameter only influences the printed units in the log file. This parameter is automatically set by examination of the user-readable energy boundary string. In the case that a single energy value is specified, the source flux is given in units of `SPI_UNIT_FLUX`. In all other cases the source flux is specified in units of `SPI_UNIT_KEV`.
- **srcXPos** [0.0]
Point source galactic longitude or Right Ascension in degrees. Only used for point source models (`type = MODEL_SOURCE`).
- **srcYPos** [0.0]
Point source galactic latitude or Declination in degrees. Only used for point source models (`type = MODEL_SOURCE`).
- **srcImage** []
- **srcEbounds** []
Contains the model energy boundaries in a `SPIBounds` object as specified by the user-readable string `ebounds`. In case that this string is empty, the `SPIBounds` object will also be empty.
- **grp_pt** [DSG_NO]
Specifies the data-space grouping type for the pointing dimension. Possible options are: `DSG_NO` (no grouping), `DSG_POINT` (one parameter per pointing), `DSG_ORBIT` (one parameter per orbit), and `DSG_DATE` (one parameter per date interval).

- `grp_det` [DSG_N0]
Specifies the data-space grouping type for the detector dimension. Possible options are: `DSG_N0` (no grouping), `DSG_DETE` (one parameter per pseudo-detector), and `DSG_EVTCLASS` (one parameter per event class).
- `grp_ebin` [DSG_N0]
Specifies the data-space grouping type for the energy dimension. Possible options are: `DSG_N0` (no grouping), and `DSG_EBIN` (one parameter per energy bin).

4 Using spi_fitlib

4.1 SPIFit_Init

TBW

C++ interface

```
int SPIFit_Init(dal_element *group,           /* input */
               SPIData      *data,          /* input */
               SPIData      *sky,          /* input */
               SPIData      *bgm,          /* input */
               int           verbose,       /* input */
               int           status);       /* input */
```

- ***group**
TBD
- ***data**
Specifies the data that should be fitted.
- ***sky**
Specifies all sky model components that should be fitted to the data. The number of sky model components to be fitted is extracted from the SPIData object.
- ***bgm**
Specifies all background model components that should be fitted to the data. The number of background model components to be fitted is extracted from the SPIData object.
- **verbose**
Verbosity level. If negative, the header message is not dumped into the log file.
- **status**
Specifies the current ISDC status when the function is entered. If `status != ISDC_OK` the function does nothing and returns the input status as return code.

Returned errors

- **ISDC_OK**
Fit initialisation was okay.

4.2 SPIFit_PrepareFit

TBW

C++ interface

```
int SPIFit_PrepareFit(SPIData  *data,
                     SPIData  *sky,
                     SPIData  *bgm,
                     SPIFitPar *par_sky,
                     SPIFitPar *par_bgm,
```

```

SPIFit      *h0,
SPIFit      *h1,
SPIFit_FCT  fitFunc,           /* input */
int         verbose,         /* input */
int         status);         /* input */

```

- ***data**
Specifies the data that should be fitted.
- ***sky**
Specifies all sky model components that should be fitted to the data.
- ***bgm**
Specifies all background model components that should be fitted to the data.
- ***par_sky**
Specifies the sky model component fitting parameters.
- ***par_bgm**
Specifies the sky model component fitting parameters.
- ***h0**
TBD.
- ***h1**
TBD.
- **fitFunc**
TBD.
- **verbose**
Verbosity level. If negative, the header message is not dumped into the log file.
- **status**
Specifies the current ISDC status when the function is entered. If `status != ISDC_OK` the function does nothing and returns the input status as return code.

Returned errors

- **ISDC_OK**
Fit preparation was okay.

4.3 SPIFit_DoFit

TBW

4.4 SPIFit_Filter

TBW

4.5 SPIFit_ScaleModels

TBW

4.6 SPIFit_GetSkyPars

TBW

4.7 SPIFit_GetBackPars

TBW

4.8 SPIFit_GetBounds

TBW

4.9 SPIFit_GetTime

TBW

4.10 SPIFit_DumpFitStatus

TBW

4.11 SPIFit_DumpFitPar

TBW

4.12 SPIFit_DumpSkyPars

TBW

4.13 SPIFit_DumpBackPars

TBW

4.14 SPIFit_DumpTime

TBW

4.15 SPIFit_WriteFitResults

TBW

4.16 SPIFit_DumpResiduals

TBW

4.17 SPIFit_DumpChisq

TBW

4.18 SPIFit_model

TBW