Astronomical Data Analysis Software and Systems X
ASP Conference Series, Vol. 238, 2001
F. R. Harnden Jr., F. A. Primini, and H. E. Payne, eds.

The FITS Embedded Function Format

Arnold H. Rots, Jonathan C. McDowell, X. Helen He, Peter E. Freeman

Harvard-Smithsonian Center for Astrophysics, 60 Garden Street MS 81, Cambridge, MA 02138

Michael Wise

Massachusetts Institute of Technology, Center for Space Research

Abstract. We have developed a format convention that allows one to specify an n-dimensional function in a FITS binary table, the FITS Embedded Function (FEF). The format allows for enumerated values, constants, and analytical functions, and arithmetic combinations of those three. The parameters of the analytical functions may, again, be enumerated, constant, or function values. The concept is intended to allow the user to extract a multi-dimensional subimage from a FEF in the same way one would extract a subimage from a primary array or image extension. The format is extremely versatile and has many potential applications. Developing a generic FEF extractor is very challenging but will allow very cost-effective reuse.

1. Scope

The FITS standard provides for the definition of n-dimensional images of data points, but in certain situations it can be extremely useful to have the ability to store an image in parameterized form, as an arbitrary function of n parameters (coordinate axes). Just as one can extract a sub-image from a primary HDU or an image extension, one should be able to extract an n-dimensional sub-image of enumerated values from an extension containing such a parameterized image.

In cases where the parameterization is possible, this representation effects a huge savings in storage space. We have developed a convention that accommodates arbitrary function specifications using FITS binary tables. We have called it the *FITS Embedded Function* (FEF).

As an example, within the Chandra X-ray Center we have applied the FITS Embedded Function to the Chandra response matrix specification. As a result, a generic FEF image extractor will be able to construct response matrices using the FEF response matrix definition.

2. Definitions

Function tables are identified by:

```
HDUCLASS= 'ASC     '
```

```
HDUCLAS1= 'FUNCTION'
```

The function is defined by:

```
FUNCTION= '<expression>'
FUNCNAME= '<name>'
```

The function expression may contain the arithmetic operators $+$, $-$, $*$, $/$, $**$, and five types of operands or "parameters:"

- FTYPEi: axes of the function evaluation space; these may just be specified as a range or may also appear as a table column (TTYPEj) if other parameters are to be enumerated along such an axis.
- DTYPEi: constants; the name is provided by DTYPEi, the value by DVALi, and the units by DUNITi. This is the "constant column."
- TTYPEi: parameters enumerated in table column i.
- VTYPEi: function components; the name is provided by VTYPEi, the function expression by VFUNCi. This is the "virtual function column."
- WTYPEi: arithmetic components; the name is provided by WTYPEi, the arithmetic expression by WFUNCi. This is the "virtual arithmetic column."

Except for the first, these operands can be thought of as columns in the table (real columns, constant columns, and virtual columns), and the function would thus be an arithmetic combination of these "columns." One should be warned, though, that operands may be defined in terms of other operands; hence reality is more complicated than this simplified view.

The assumption is that implementation of a FEF reader will include the following functionality.

```
typedef struct {
  char name[32] ;
  double min ;
  double max ;
  int num ;
} FefParam ;

FefParam parms[n] ;
fits_file fef ;

double* image = readFef (fef, n, parms) ;
```

`readFef` returns an n-dimensional image where the lengths of the axes are set by `parms[i].num`, the names by `parms[i].name`, and the minimum and maximum by `parms[i].min` and `parms[i].max`, respectively. Each `parms[i].name` must correspond with an FTYPE value and the minimum and maximum values must not exceed the corresponding FLMIN and FLMAX values. n should be equal to FAXIS.

Extraneous columns are allowed. However, one should take care when filtering on such a column (or any other, for that matter), that such filtering can only be guaranteed to yield a valid FEF extension if all but one of the FTYPE variables contains more than one point. Even then, it will still require that all FAXISi keywords are corrected.

3. The Coordinate Axes

The evaluation space of the function (or the image coordinates, if you like) is set by a set of "pseudo coordinate axis" keywords:

- FTYPEi:
- FAXIS: The number of coordinate axes in evaluation space
- FAXISi: The number of points along axis i (for enumerated axes only)
- FTYPEi: Name of axis i
- FLMINi: Legal minimum values for axis i (required for free-running variables)
- FLMAXi: Legal maximum values for axis i (required for free-running variables)

One should distinguish between enumerated and free-running variables (coordinates). The free-running variables only appear as parameters in function expressions while for enumerated variables the function value (or some parameter value) is explicitly given for a list of the variable's values. As an extreme example, a measured image could be represented with one row per pixel, a column for each coordinate, and one for the image value, where the coordinate position of each pixel would be explicitly given on each line: all coordinates would be enumerated. If, on the other hand, the function were a one-dimensional Gaussian, there would only be one free-running variable. Somewhere in between, one could conceive of a function in x and y, where the function value is a one-dimensional Gaussian with free-running variable x, but where the width of the Gaussian is enumerated for all values of y.

FLMIN and FLMAX are clearly needed for free-running variables to define the range over which the function is defined.

We shall assume that if function values are requested for values of enumerated variables that fall between the enumerated grip points, the values of all enumerated columns will be interpolated linearly. It is important to note that *the parameter values are interpolated, not the function values.*

If the values of an entire row are to be held constant for a range of values of an enumerated variable, one may specify bins by using a <FTYPE>_LO and <FTYPE>_HI column.

In some cases (sparse images, e.g., response matrices) it is desirable to limit the domain of a free running variable depending on the values of the enumerated variables. This may be done by incorporating two columns for FDMINi and FDMAXi.

4. FUNCTION Specification

The FUNCTION definition is an arithmetic expression in which the operands may be the *values* of any of the following: FTYPEn, DTYPEn, TTYPEn, VTYPEn, and WTYPEn, with operators $+$, $-$, $*$, $/$, $**$, and parentheses allowed, as in:

```
FUNCTION= 'Norm - Scale * (X2 + Y2)'
FUNCNAME= 'HRMA_EffArea'
BUNIT   = 'mm**2   '
```

5. VTYPE

Virtual function columns (VTYPEi) are defined through functions:

```
VFUNCi = 'G(P1, P2, P3, ...; C)'
```

where G is chosen from a defined set of function names and where P_i may be the value of any valid operand—one of the following: FTYPEn, DTYPEn, a constant number, TTYPEn, VTYPEn, or WTYPEn. C is the name of a parameter object or a coefficient object that is specific to the function G and whose attributes need to be specified as $C\_ < name >$; not all functions require a parameter object. For each parameter object attribute there has to be one TTYPE, DTYPE, VTYPE, or WTYPE that carries its name as value. VFIELDS specifies the number of VTYPEs that are defined.

Note that allowing VTYPEs and WTYPEs to be used in the definition of VTYPEs provides for the specification of nested functions. This is a powerful capability that requires particular care to prevent recursion.

6. WTYPE

In general, arithmetic virtual columns (WTYPEi) are defined as:

```
WFUNCi = 'P1 ^ P2 ^ ...'
```

Where "^" denotes an arithmetic operator $+$, $-$, $*$, $/$, $**$, and parentheses are allowed. P_i may be the value of any of the following: FTYPEn, DTYPEn, a constant number, TTYPEn, VTYPEn, or WTYPEn. WFIELDS specifies the number of WTYPEs that are defined.

Acknowledgments. This project is supported by the Chandra X-ray Center under NASA contract NAS8-39073.