.pl 60
.mb 3
.. fare ^OR 72

The revised spectral form package
Programmers' notes

Software and documentation prepared by:
L.Chiappetti - IFC Milano

31 August 1987

1. Introduction

This document describes the main features of the revised spectral form package, intended for use with all Exosat and IUE programs. This document is not aimed to the general user, which will find most of the information needed to **RUN** programs in the appropriate documentation (see Refs. 2 and 5). This is instead an explanation for the advanced user or programmer which wants to develop new programs, or maintain/modify/customize existing programs. Note that, as the source code of the various routines is adequately documented, the reader is invited to look at the comments in the code for most details.

1.1 Motivations

The revised spectral form package here described replaces a set of similar subroutines available in miscellaneous formats for use with the original IUE data analysis package (see ref. 1) or in library format (in file LUCIOLIB TXTLIB) for use within the Exosat data analysis package (see ref. 2, section 5.9 etc.).

Some of the motivations inspiring the revised spectral form package were already present in the earlier versions (see 1.2 below), however one further, basic motivation for the current refurbishing of the whole package was the elimination of all conflicts between the previous IUE and Exosat versions (see e.g. ref. 3).

The remaining motivations which led to the present form

of the spectral package are :

Provide a library of the commonest spectral form in UV and X-ray astronomy, in a way which supports different units for both flux and energy/wavelength/frequency, and provides normalization constants and other spectral parameters in an "universal" (i.e. unit-independent) format.

Provide an easy programmer interface (through the use of "distribution" routines), which conceals the details in the individual routines (communicating through common blocks), and improves the legibility of the program. The routines may be used as building blocks for new programs.

Provide easy maintenance of both existing and new programs (e.g. allowing addition of new spectral forms, or alteration of parametric values) with minimal changes to the code (only recompilation will be needed for most programs).

1.2 History

The need for a spectral package arose originally for the IUE data analysis system (see Ref. 1). In particular the requirements for unit-independence (see also Ref. 4) originated at the time when a spectral fitting package was required for IUE data (inspired by previous, undocumented, Ariel-5 software). The original IUE fitting program (never incorporated in the "official" documentation) supported many of the features in the present package, like a menu of available spectral forms and parameters to be fitted etc., a distribution routine to keep track of which parameters to fit or keep fixed etc. Of course this program fitted erg/cm^2/s/A vs A. Universal constants were kept out of the normalization factors (which were therefore unit-independent).

The same spectral package was partially modified and complemented by other routines for use in the Exosat ME data analysis system (see Ref. 2, chapter 5). The set of programs using the package (then contained in libraries LUCIOLIB and MELIB - the latter for the parts

relevant to X-ray data only) included a convolution (spectra simulation) program, fitting, chi-square grid and flux integration. Of course all programs worked in photons/cm^2/s/keV vs keV. It was also found more convenient to include universal constants in normalization factors (the numbers are smaller, which is computationally advantageous).

These and other little differences between the two packages, as well as the absence of a chi-square grid program for UV spectra, made the time ripe for a complete refurbishing of the package, which is now totally independent of wavelength range, and will be used both by the X-ray (Ref. 2) and UV (Ref. 5) programs.

Although the earlier versions of the package were presented in a less formal way, the current version will be officially designated as Version 3 (the original IUE and Exosat versions are referred as version 1 and 2 respectively).

To avoid conflict with earlier versions, and continue support of the same, all subprogram names (and common block names) have been changed (see comments in source listing).

.cp53

2. Global scheme

2.1 Implementation notes

The current version of the spectral form package is implemented in IBM VS Fortran for use under VM/SP (CMS). However, since most of the i/o is done outside the package in the calling program, and since great attention and care has been given to portability, the package is virtually machine-independent (only minor modifications needed), and could be usefully recycled in the future.

The source code is available as part of the file LUCIOLIB LIBRARY, the relocatable code is available in LUCIOLIB TXTLIB respectively on disks A2 and B2 of the EXOSAT VM. Instructions for library maintenance (single member compilation etc.) for IBM users can be found in Ref. 6. Besides standard Fortran library externals, reference is made, by a small number of individual spectral form routines, to numeric integration routines in the IBM

Scientific Subroutine Package (system file FORTSSP TXTLIB). When loading a program or linking a module, **make sure that libraries LUCIOLIB and FORTSSP are available in your GLOBAL TXTLIB command.**

One of the major changes in Version 3 is the use of INCLUDE files for all common blocks, and most parameter statements. This means that the original code for such pieces of program are stored in a single file, and are included at compilation time where needed. This avoids typing or copying errors in updating. Details about the use of INCLUDE on IBM (such statement is Fortran 77 standard, however subtle differences in implementation on various machines exist) can be found in Refs. 6 and 7. The source code for all INCLUDEs used in the spectral package is file LUCIOMAC COPY on disk A2 of the EXOSAT VM. **When compiling any (sub)program which makes reference to such INCLUDEs, the macro library LUCIOMAC MACLIB (on same disk) shall be available** : this is obtained issuing once the CMS command:

GLOBAL MACLIB LUCIOMAC

2.2 Composition and structure of the package

The structure of the package is shown in Fig. 1. The circles indicate common blocks. Such common blocks are filled by calls to dedicated routines (arrows entering the circles). All other routines (rectangles) access (some of) such common blocks (as shown by lines without arrows). Lines connecting the various rectangles indicate which routines are called by which (arrows enter the CALLED routine).

There are two main common blocks : SPECMN (partially replacing version 2 SPECOM) is accessed by most programs and holds basic things (like the choice of units); FFLAGS (replacing version 1 and 2 FLAGS) is used by fitting programs only (more exactly by programs using the sum of more spectral forms).

.cp4

Auxiliary common blocks are SPEPAR (partially replaces version 2 SPECOM), which contains the parameters for programs using a single spectral form; and SPEHCD and SPUVAB, which contain the information relevant to

interstellar hydrogen (X-ray) or dust (UV) absorption respectively. Note that fitting programs find such information in common FFLAGS instead !

The core of the package is a set of individual spectral forms, listed in Table 1. Each spectral form corresponds to 2 or 3 entry points in a single routine. Each routine always has an information entry point (INFOnn), used by the calling program to derive information like the name of the spectral form, the number of parameters, and their names; and SPEnn, which calculates the flux. Sometimes an additional entry point, alias to SPEnn, is provided for legibility (e.g. SPE01 is also called POWLAW, SPE02 is also called BLACK etc.).

Note that when compiling with COMPLIB (Ref. 6) some precautions are necessary. In particular library portions may consist of more than one subroutine which are relocated together (this is the case of SOMMP2 and the related routines), or of more entry points. Generally the name known to COMPLIB is the same name of the first entry point (the one with SUBROUTINE or FUNCTION statement and not with ENTRY statement). Exceptions are the following: the routine SPECS must be first deleted from library with **TXTLIB DEL LUCIOLIB SPECS** and then reloaded with **COMPLIB LUCIOLIB SPECS1** (the member name is SPECS); also all spectral form routines must be first deleted with **TXTLIB DEL LUCIOLIB SPE**nn and then reloaded with **COMPLIB LUCIOLIB** aliasname (aliasnames are e.g. POWLAW for SPE01 etc. as in Table 1; the member name is SPEnn).

Most of such routines are functions of the form y=f(x,A), where A is the array of spectral parameters (normalization, temperature or spectral index, etc.), x is energy (or wavelength or frequency) and y is monochromatic flux in the appropriate unit (see 2.3 below).

An additional set of routines is represented by distribution routines. There are three basic distribution routines, plus the specialized versions for X-ray and UV. One distribution routine is an information call (routine INFO), at all similar to the INFOnn mentioned above. The other two (respectively SPECS and SPESUM) are calculation calls of the form y=f(x,A), where f is either a single spectral form, or the sum of a number of different spectral components. Distribution routines provide transparent access to the selected spectral form(s).

The specialized versions (either for single or summed component(s)) call the respective basic version, then correct the flux for the relevant gas (X-ray) or dust (UV, optical) absorption. The absorption routines are not part of this package, however their calling sequence is recalled in 6.1 below for completeness.

A third set of routines is represented by menu routines used by the fitting and chi-square grid programs. They have been designed in a way that a single routine could be used for both families of programs and in all wavebands. The main menu routine is READP (an additional entry point SOMMP provide a summary of selected parameter), while specialized versions handle again gas or dust absorption. This set of routines is used to fill or access the common block FFLAGS, used by SPESUM to handle the sum of several components. The interfacing with Bevington's fitting routine CURFIT (see Ref. 8) is outside the scope of this package, however the essentials are recalled in 6.2 below.

Table 1

Code	Spectral form	Call as	Alias	Info call	
1	Power law	SPE01	POWLAW	INFO01	see note 1
2	Blackbody	SPE02	BLACK	INFO02	
3	Bremsstrahlung	SPE03	THERM	INFO03	see note 2
4	Exponential	SPE04	EXPON	INFO04	
5	Bremsstrahlung	SPE05	THBNW	INFO05	see note 2
6	"Boltzmann"	SPE06	BOLTZM	INFO06	
7	Thick disk (Bath)	SPE07	BDISK	INFO07	
8	Gaussian line	SPE08	GLINE	INFO08	see note 1
9	Spare	SPE09	---	INFO09	see note 3
10	Compton	SPE10	COMPT	INFO10	
11	Czerny mod. BB	SPE11	CZERNY	INFO11	
12	Twin power law	SPE12	TWINPL	INFO12	see note 1
13-15	Spare	SPEnn	---	INFOnn	see note 3

Notes:

1) The spectral forms 1 (power law), 8 (gaussian line) and 12 (twin power law, i.e. two power law with a break) are pure analytical expressions in the chosen units (therefore normalization, spectral index and other parameters are dependent on unit selection, i.e. if dN/dE is selected,

spectral index is photon index, break is in keV etc., but if Fl is selected, index is wavelength index and break is in A, etc.).

2) The form 3 is a thermal Bremsstrahlung with approximated Gaunt factor (according to Ref. 4). The form 5 (based on ESOC code) uses a more complex expression of the Gaunt factor (Kellogg, Baldwin and Tucker), and combines 100% of a thermal spectrum with Z=1 with 34% of one with Z=2.

3) Spare forms 9 and 13-15 are available for customization (see 5. and 7. below). Form 9 is defined "spare for edges" for consistency with older documentation (see Ref. 2 section 5.9)

For details refer to the source listing, and to the formulae in Appendix.

.pa

2.3 Sample usages

The following examples indicate how to make use of the spectral routines to build typical programs. As a general rule one shall :

First of all select whether one wants dN/dE (photon/cm^2/s/keV vs keV), or Fl (erg/cm^2/s/A vs A) or Fn (erg/cm^2/s/Hz vs Hz), whether temperatures are in keV or Kelvin, and whether physical constants are included or not in the normalization. This can be "hardwired" in the program, or subject to user choice. (See 3.1 below).

For X-ray programs one shall select the cross section code for absorption. (For UV programs the model, when more than one model will be supported, currently only Seaton's models is supported). (See 3.2/3 below).

Then one selects the spectral form(s) to be used (see 3.1 and 4.4 below; if necessary the name of the parameters can be obtained with an info call, see 4.1), and the parameter values (see 3.4 or 4.4), finally one does all calculations (see 4.2/3 and 5.).

The details on the calling sequences are given in 3. to 5. below. For more details refer to the listing of the individual routines.

Instructions to customize a program, adding new spectral forms, or modifying the existing ones, are given in section 7.

2.3.1 A single-form, indirect program

The scheme here described for a program calling a single-component spectral form indirectly (i.e. through the distribution routines) is followed e.g. by the Exosat programs INTEGRA and CONVOLVE (in their latest version), to which one is referred for details. One example for an X-ray case is shown, but changes should be obvious.

```
optionally ask user for units etc.

IF(SPUNIT('dN/dE'   ))GOTO error        select units
IF(SPTEMP('keV'     ))GOTO error        select temperature
units
IF(SPCONS('Included'))GOTO error        include physical
constants

(optionally) ask for cross-section      (or model for UV)
IF(SPCROS(code))GOTO error              (SPMODL for UV)

ask user for desired NH                 (or AV for UV)
IF(SPNH(NH))GOTO error                  (SPAV for UV)

ask user for spectral form code
IF(SPCODE(code))GOTO error              select spectral form

CALL  INFO(code,N,NAME,PARA)            get how many
parameters etc.
```

.cp3

```
ask the values for the parameters (called PARA(i))
and read in N values in array A

CALL SPARGS(A,N)                        set up parameters
(if using
                                        SPECSX or SPECUV
                                        below)
```

Do the calculation (ev. within a loop or other subroutine) via

```
Y=SPECSX(X)                             or
Y=SPECUV(X)                             for UV, or
Y=SPECS(X,A)                            if unabsorbed
```

2.3.2 A single-form, direct program

The scheme here described is for a program calling a

single-component spectral form directly (i.e. the program only handles a particular spectral form, e.g. a power law). One example for an X-ray case is shown, but changes should be obvious.

optionally ask user for units etc.

```
IF(SPUNIT('dN/dE'   ))GOTO error        select units
IF(SPTEMP('keV'     ))GOTO error        select temperature
units
IF(SPCONS('Included'))GOTO error        include physical
constants

The nn-th form is desired
CALL  INFOnn(N,NAME,PARA)               get how many
parameters etc.
```

ask the values for the parameters (called PARA(i))
and read in N values in array A

Do the calculation (ev. within a loop or other subroutine) via

```
Y=SPEnn(X,A)                            value is unabsorbed
```

Any eventual absorption is care of the user

2.3.3 A multi-component program

The scheme here described for a program calling a multi-component spectrum is followed e.g. by the FIT and GRID programs. One example for an X-ray, fitting, case is shown, but changes should be obvious.

optionally ask user for units etc.

```
IF(SPUNIT('dN/dE'   ))GOTO error        select units
IF(SPTEMP('keV'     ))GOTO error        select temperature
units
IF(SPCONS('Included'))GOTO error        include physical
constants
```

All dialogue on spectral forms, parameters etc. is concealed to the user in the menu routines below :

.cp4

```
CALL READNH(A,DELTAA,NP,'FIT')          (READAV for UV)
CALL READP (A,DELTAA,NP,'FIT')          this does all here !
```

```
CALL SOMMP(XXX,'FIT',YYY)                    if printed summary
desired

Do the calculation (ev.  within a loop or other subroutine)
via (in a fitting program this is done in FUNC)

Y=SPEX2(X,A)                                 or
Y=SPEUV2(X,A)                                for UV, or
Y=SPESUM(X,A)                                if unabsorbed
desired

At the end if a summary of unit conversion is desired
CALL SOMMP2(A)
```

3. Common handling routines

3.1 The main common SPECMN

Common block: SPECMN
Include file: SPECMN (member in LUCIOMAC MACLIB)

Common content:

Type	Variable	Initialized to:
Integer	IUNIT	1
Logical	TKEV	TRUE
Logical	PHYSIC	TRUE
Integer	MAXFRM	15
Integer	ISPCOD	0

Initialized by: Block data, relocated with SPUNIT

The IUNIT flag assumes the values 1,2,3 respectively for fluxes in erg/cm^2/s/A vs A, erg/cm^2/s/Hz vs Hz and photon/cm^2/s/keV vs keV.

The TKEV flag is .TRUE. is temperature is to be measured in keV, .FALSE. if it is to be measured in Kelvin.

The PHYSIC flag is .TRUE. if physical constants are included in the normalization factor (as done in the Exosat fitting programs, and in the latest IUE fitting program, contrary to the older IUE program), .FALSE. otherwise. See Ref. 4 for details.

MAXFRM is the maximum number of different spectral forms in the package, currently FIXED to 15.

ISPCOD is the code of the spectral form currently selected

(default is 0=none). See Table 1 for codes. Used by single spectral form programs only.

.cp5
3.1.1 Set up routines for SPECMN

The following calls are used to set up the flags in SPECMN. They are all entry points of a single routine SPUNIT (which also contains the block data necessary to initialize the common). All entries are FUNCTIONs of LOGICAL (tout court) type, which return a .TRUE. value in case of error. The general calling sequence is therefore :

IF routine(keyword) GOTO error-label

The selection of the flag value is operated through the keyword or code. Keywords shall be typed exactly as indicated (within quotes, lower case, capital initial). Typing error, or any other keyword, or code values out of range, will cause the routine to assume the error value. The following entry points are available :

Purpose: **select units for flux and energy etc.**
Calling sequence: IF(SPUNIT(keyword))GOTO error-label

Allowed keywords:

'Flambda'	for erg/cm^2/s/A	vs A
'Fnu'	for erg/cm^2/s/Hz	vs Hz
'dN/dE'	for photon/cm^2/s/keV	vs keV

Purpose: **select units for temperature**
Calling sequence: IF(SPTEMP(keyword))GOTO error-label

Allowed keywords:

'keV'	for temperature in keV
'Kelvin'	for temperature in Kelvin

Purpose: **select whether physical constants are included in normalization factors**
Calling sequence: IF(SPCONS(keyword))GOTO error-label

Allowed keywords:
'Included'
'Notincluded'

Purpose: **select spectral form**
Calling sequence: IF(SPCODE(code))GOTO error-label

Allowed codes : 1 to MAXFRM (currently 1 to 15)

A call to SPCODE is needed only if the spectral form is not selected otherwise.

.cp7
3.2 The hydrogen column density common SPEHCD

Common block: SPEHCD
Include file: SPEHCD (member in LUCIOMAC MACLIB)

Common content:

Type	Variable	Initialized to:
Real	CNH	0.0
Integer	ICROS	5
Real	POXY	1.0
Real	PFE	1.0

Initialized by: Block data, relocated with SPNH

The ICROS flag assumes the values 0 to 5 to select different models for the photoelectric absorption cross section (see 6.1 for details)CNH is the hydrogen column density in 10^{22} cm^{-2}. (note that some routines ignore ICROS and CNH and use similar information in common block FFLAGS)

POXY and PFE are the relative abundances of Oxygen and Iron (with respect to cosmic abundance). They are currently set to 1.0 (cosmic abundance) FIXED.

3.2.1 Set up routines for SPEHCD

The following calls are used to set up the values in SPEHCD. They are all entry points of a single routine SPNH (which also contains the block data necessary to initialize the common). All entries are FUNCTIONs of LOGICAL type, which return a .TRUE. value in case of error. See 3.1.1 for details. The general calling sequence is therefore :

IF routine(keyword) GOTO error-label

The selection of the flag value is operated through the keyword or code.

The following entry points are available :

Purpose: **select value of column density**
Calling sequence: IF(SPNH(value))GOTO error-label

Allowed values : Any non-negative value

Purpose: **select cross section code**
Calling sequence: IF(SPCROS(code))GOTO error-label

Allowed codes : 0 to 5

Purpose: **select oxygen abundance**
Calling sequence: IF(SPOXY(value))GOTO error-label

Allowed values : Any. Call is currently dummy

Purpose: **select iron abundance**
Calling sequence: IF(SPFE(value))GOTO error-label

Allowed values : Any. Call is currently dummy

.cp9
3.3 The UV extinction common SPUVAB

Common block: SPUVAB
Include file: SPUVAB (member in LUCIOMAC MACLIB)

Common content:

Type	Variable	Initialized to:
Real	AV	0.0
Integer	IMODEL	1

Initialized by: Block data, relocated with SPAV

The model code IMODEL is currently not used

AV is the visual extinction in magnitudes. (note that some routines ignore IMODEL and AV and use similar information in common block FFLAGS)

3.3.1 Set up routines for SPUVAB

The following calls are used to set up the values in

SPUVAB. They are all entry points of a single routine SPAV (which also contains the block data necessary to initialize the common). All entries are FUNCTIONs of LOGICAL type, which return a .TRUE. value in case of error. See 3.1.1 for details. The general calling sequence is therefore :

IF routine(keyword) GOTO error-label

The selection of the flag value is operated through the keyword or code. The following entry points are available :

Purpose: **select value of visual extinction**
Calling sequence: IF(SPAV(value))GOTO error-label

Allowed values : Any

Purpose: **select extinction model code**
Calling sequence: IF(SPMODL(code))GOTO error-label

Allowed codes : Any. Call is currently dummy.

.cp5
3.4 The parameter common SPEPAR

Common block: SPEPAR
Include file: SPEPAR (member in LUCIOMAC MACLIB)

Common content:

Type	Variable	Initialized to:
Real	A(NFPARM)	No value

Initialized by: Not initialized

.cp3
The array A contains the spectral parameters for the currently selected spectral form (see SPCODE above, used only by the same programs calling SPCODE). The maximum dimension of A is parametric (see 7.1 below).

3.4.1 Set up routines for SPEPAR

Purpose: set up parameter values
Calling sequence: CALL SPARGS(B,N)

Arguments : B is user-generated array of parameters
N is the number of parameters (the

dimension of B shall be at least N)

The user supplied parameters, as necessary for the current spectral form, will be copied to array A in common SPEPAR.

3.5 The multi-component (fitting) common FFLAGS

Common block: FFLAGS
Include file: PARACP and FFLAGS (members in LUCIOMAC MACLIB)

Common content:

Type	Variable	Initialized to:
Integer	NSP	0
Integer	JSPCOD()	all 0
Integer	IC()	all -1
Integer	NC()	all 0
Integer	NHIC	-1
Integer	NHNC	0
Integer	JCROS	depending on program
Integer	INDARR()	not initialized
Real	B()	all 0.0
Real	UNIT()	all 1.0
Real	ENH	0.0
Real	UNH	1.0E21 (sic!)
Real	STEP()	not initialized
Character	NHORAV	not initialized

.cp5

Initialized by: Block data, relocated with READP NHORAV and JCROS are set up by a call to READNH or READAV (see 4.4.2/3 below).INDARR and STEP are set up by calls to SETP (in grid programs only).

NSP is the number of different components currently defined (maximum is parametric, see 7.1 below)

JSPCOD is an array (size is the maximum defined above) containing the codes of all defined spectral components.

.cp4

B is a matrix (size is parametric, see 7.1), where B(i,j) contains the j-th parameter of the i-th component.IC is a matrix of the same size as B which contains a flag, indicating whether the corresponding B(i,j) is either undefined (value -1),, fixed (value 0), fitted (value 1) or stepped (value 2, chi-square grid programs only).NC is a matrix of the same size as above, which, if IC(i,j) is

1, contains the address of the parameter in the array of fitted parameters used by CURFIT.UNIT is a matrix of the same size as above, containing the units (scaling factors) for all parameters in B. The values in B, when fitted with CURFIT, shall be kept small. The option of assigning values to UNIT is currently unused (all factors equal to 1.0).

ENH, NHIC, NHNC and UNH have the same role as B, IC, NC and UNIT, but for the absorption parameter. This is the hydrogen column density for X-ray programs and the visual extinction for UV programs. Note that UNH is initialized to 10^{21}, which is suitable for **X-RAYS ONLY.** The value of UNH is reset to 1.0 (as necessary for UV programs) when calling routine READAV.

JCROS contains the cross section code (same as 3.2) for X-ray programs, or the model code (same as 3.3) for UV programs. This is the value actually used by fitting programs, not the one in SPEHCD or SPUVAB.

NHORAV is a 16-character string containing the name of the absorption parameter. Note that using a character variable in a common together with non-character variables is an extension to standard Fortran 77.

INDARR is a (2,2) matrix containing the pointers to row and column in arrays B etc. for stepped parameters (grid programs only).STEP is an array containing the step value for stepped parameters (grid programs only).

3.5.1 Set up routines for FFLAGS

See menu routines READP, READNH and READAV in 4.4 below.

.cp50

4. Distribution routines

4.1 Information call

Purpose: obtain info on spectral form
Calling sequence: CALL INFO(ICODE,M,NAME,PARNAM)

Argument ICODE : is the code (see Table 1) of the spectral form for which info is desired. If ICODE is 0 and a spectral form has been selected (see SPCODE above),

it is set on return to the code actually selected.

Argument M : (returned) is the number of parameters requested by this spectral form. If ICODE was 0 and no valid spectral form was yet selected, M is set to 0.

Argument NAME : (returned) is a 16-character string containing the name of this spectral form. If ICODE was 0 and no valid spectral form was yet selected, NAME is the string 'Unsupported'.

Argument PARNAM : (returned) is an array of 16-character strings (at least M) containing the names of the parameters of this spectral form. If ICODE was 0 and no valid spectral form was yet selected, PARNAM is not set.

4.2 Single form distribution calls

All these calls are REAL FUNCTIONs, most of the form y=f (x,A), some of the form y=f(x). The latter form has been used where more convenient.

4.2.1 Unabsorbed form

Purpose: compute monochromatic flux in units as set up by a call to SPUNIT, see 3.1.1)

Calling sequence: Y=SPECS(X,A)

Argument X : is energy, wavelength or frequency.
Argument A : is the array of spectral parameters

Returned value : is the monochromatic flux for the single spectral form currently set up (with SPCODE or directly, see 3.1.1)

Note : When compiling/replacing this member in library, it shall be called as SPECS1. Member SPECS shall be removed from library in advance, with "manual" usage of the TXTLIB DEL command.

.cp50

4.2.2 X-ray form

Purpose: compute absorbed monochromatic flux.

Calling sequence: Y=SPECSX(X)

Argument X : as 4.2.1

Returned value : is the monochromatic flux computed by SPECS (see 4.2.1 above), absorbed for current column density (set up with SPNH etc. see 3.2.1) via the routine ATTEN (see 6.1) The parameters shall have been set up with a call to SPARGS (see 3.4.1).

4.2.3 UV form

Purpose: compute absorbed monochromatic flux

Calling sequence: Y=SPECUV(X)

Argument X : as 4.2.1

Returned value : is the monochromatic flux, similarly to 4.2.2 above, but absorbed for current extinction (set up with SPAV etc. see 3.3.1) via the routine DERED (which is in library IUEAUXLB) (see 6.1)

4.3 Summed form distribution calls

All these calls are REAL FUNCTIONs, of the form y=f (x,A). They return the monochromatic flux of a sum of spectral components. The maximum number of spectral components allowed is parametric (see 7.1) and is currently 4. (Note that earlier up to 10 components were allowed, but such a number has never been used).

4.3.1 Unabsorbed form

Purpose: compute monochromatic flux in units as set up by a call to SPUNIT, see 3.1.1)

Calling sequence: Y=SPESUM(X,A)

Argument X : is energy, wavelength or frequency.
Argument A : is the array of fitted spectral parameters, with the latest values of the current fitting iteration. All other parameters are obtained internally from array B of common FFLAGS.

.cp5

Returned value : is the monochromatic flux for the sum of the components currently set up (with READP)

4.3.2 X-ray form

Purpose: compute absorbed monochromatic flux

Calling sequence: Y=SPEX2(X,A)

Arguments : same as 4.3.1

Returned value : is the monochromatic flux computed by SPESUM absorbed for current column density via the routine ATTEN (see 6.1). The current column density and cross section are taken from common FFLAGS, only the oxygen and iron relative abundances are taken from SPEHCD (currently the relative abundances are set to 1.0, that is cosmic abundance).

Note : For fitted parameters only, there is a limit of 10^{19} atoms/cm^2 for the hydrogen column density. Lower values are not allowed (no such limitation if the parameter is not fitted).

4.3.3 UV form

Purpose: compute absorbed monochromatic flux

Calling sequence: Y=SPEUV2(X,A)

Arguments : same as 4.3.1

Returned value : is the monochromatic flux computed by SPESUM absorbed for current extinction via the routine DERED in library IUEAUXLB (see 6.1). The current extinction and model are taken from common FFLAGS.

Note : For fitted parameters only, there is a limit of 0.01 for the visual extinction. Lower values are not allowed (no such limitation if the parameter is not fitted).

.cp50

4.4 Menu and summary calls

All these calls are intended for use with fitting and chi-square grid programs, although they could be used for

any programs dealing with the sum of more spectral components. The menu calls are used to fill common FFLAGS (see 3.5), while summary calls access such common in order to display its content. Note these calls contain terminal i/o : portability is not impaired by use of the User-Program Communication package (see Ref. 9).

4.4.1 Main (no absorption) menu call

Purpose: handle all dialogue to set up the various spectral components for fitting programs.

Calling sequence: CALL READP(A,DELTAA,NPAR,keyword)

Argument A : (returned) is the array of fitted spectral parameters, with the initial guess values. All other parameters are stored internally in array B of common FFLAGS.
Argument DELTAA : (returned) is the array of increments for fitted parameters (to be used in CURFIT)
Argument NPAR : (returned) is the number of fitted parameters

Allowed keywords: 'FIT' or 'GRID' (used to control the dialogue and program flow; any other value cause impredictable results. Type all in upper case.

Externals : INFO and SETP in same library
READIN in User-Program Communication package

Note that this routine, as well as the two routines below, handle the ultimate choice whether a parameter has to be fitted, stepped (for grids) or kept fixed via an internal call to routine SETP. This routine is not intended to be called by user programs. Enough information is provided in the source listing, therefore it is not described here.

4.4.2 X-ray form

This routine shall be called prior to READP by X-ray programs, in order to fill common FFLAGS with information relevant to absorption.

Purpose: handle dialogue for hydrogen column density

Calling sequence: CALL READNH(A,DELTAA,NPAR,keyword)

Arguments : same as READP

Externals : SPCROS and SETP in same library
READIN in User-Program Communication package

.cp10
4.4.3 UV form

This routine shall be called prior to READP by UV programs, in order to fill common FFLAGS with information relevant to absorption.

Purpose: handle dialogue for extinction

Calling sequence: CALL READAV(A,DELTAA,NPAR,keyword)

Arguments : same as READP

Externals : SETP in same library
READIN in User-Program Communication package

4.4.4 Menu summary call

This call provides a printout summary of the selections made in READP (the print file is assumed on logical unit 7), and supplies additional information to the calling program. It shall be called after READP. This routine is an entry point of READP, with which it shares all character descriptors.

Calling sequence: CALL SOMMP(HEAD,keyword,HEA2)

Allowed keywords: 'FIT' or 'GRID' (as in READP)

.cp3
Argument HEAD : (returned) is an array of 16-character strings, with as many elements as fitted parameters, containing the names of the fitted parameters in the order selected.
Argument HEA2 : (returned) is an array of 2 16-character strings, containing the names of the stepped parameters (grid programs only)

4.4.5 Final summary call

This call is used (by fitting programs) to display on a print file (logical unit 7 assumed) the value of the spectral parameters in different choices of units.

Calling sequence: CALL SOMMP2(A)

Argument A : is the array of fitted spectral parameters, with their latest values. All other parameters are obtained internally from array B of common FFLAGS.

Externals : INFO and AUXn (n=1 to 6) in same library

The auxiliary calls AUXn (see listing for details), perform the following types of unit conversion:

AUX1 converts power law normalizations (see Ref. 4)
AUX2 converts power law spectral indices (see Ref. 4)
AUX3 converts energy, wavelength and frequency into each other
AUX4 converts keV into Kelvin and v.v.
AUX5 converts blackbody normalization (see Ref. 4)
AUX6 converts Bremsstrahlung normalization (see Ref. 4)

5. Single spectral form calls

The single spectral forms available are listed in Table 1. Each one is a member in LUCIOLIB TXTLIB, with two or three entry points. The member name is generally the name of the SPEnn entry point, however when compiling with COMPLIB the "alias" name shall be used, if available (the member SPEnn shall be explicitly deleted with TXTLIB DEL before reloading, see note in 2.2) Some of the routines call internally other functions, which are relocated together. One exception is the GAUNT routine, which is an independent member.

The general calling sequences are :

INFO call : CALL INFOnn(M,NAME,PARNAM)

Argument M : (returned) is the number of parameters requested by this spectral form.

Argument NAME : (returned) is a 16-character string containing the name of this spectral form.

Argument PARNAM : (returned) is an array of 16-character strings (at least M) containing the names of the parameters of this spectral form.

Normal call : Y=SPEnn(X,A)
or Y=alias(X,A)

Argument X : is energy, wavelength or frequency.
Argument A : is the array of spectral parameters

Returned value : is the monochromatic flux

The number of parameters for a spectral form may vary from 2 to a parametric (see 7.1) maximum, currently 5. Unsupported (spare) forms appear to have 1 parameter. The current parameter definitions are :

.cp15

Code	Spectral form	No. of params	and list
1	Power law	2	Normalization, Spectral index
2	Blackbody	2	Normalization, temperature
3	Bremsstrahlung	2	as number 2
4	Exponential	2	as number 2
5	Bremsstrahlung	2	as number 2
6	Boltzmann	3	Normalization, temperature, spectral index
7	Thick disk	3	Normalization, temperature and ratio of outer to inner radius
8	Gaussian line	3	Normalization, line centre, line FWHM
10	Compton	3	Normalization, electron temperature and optical thickness
11	Czerny modif. BB	3	Normalization, temperature and density
12	Twin power law	4	Normalization of first power law, spectral index of first and second power law, break position

Note that the normalization for form 5 does not include the square root of kT at denominator (contrary to forms 3 and 4), and is independent of the choice about inclusion of physical constants (always included). For form 12 the normalization of the second power law is computed internally (it is not a free parameter) and is available with a call to SOMMP2.

Concerning the internal operation of the various routines, see the listing. Any peculiarity is indicated by comments.

Note that older routines resolve internally any unit conversion with dedicated code (according to Ref. 4), while

newer routines adopt a different approach, i.e. convert the x-axis units to energy, compute fluxes as dN/dE, and then convert fluxes to the desired units. This conversion is done via an internal call to the dedicated routine MOAMED.

A further internall call to DUMFUN is done by spare spectral forms.

Instructions to build new spectral forms to be used instead of spares are given in 7.2. Among possible applications, one should note :

The possibility of building a form (copied from one already existing) with an intrinsic absorption (one parameter more)

The possibility of building a form (copied from one already existing) with an high energy cutoff, or an edge (since cutoffs and edges are multiplicative, and not additive, it has been chosen not to consider them as independent spectral forms).

.cp50

6. External interfaces

6.1 Interstellar absorption packages

These sets of routines are not part of the present package. The calling sequence of the (outer level) routines is recalled.

6.1.1 X-ray photoelectric absorption

Calling sequence: F=ATTEN(EE,ANH,ICROS,POXY,PFE)

Argument EE	:	is energy in keV
Argument ANH	:	is column density in 10^{22} atoms/cm^2
Argument ICROS	:	is the cross section code (see below)
Argument POXY	:	is the relative abundance of oxygen
Argument PFE	:	is the relative abundance of iron
Value F	:	is the (returned) attenuation factor
Externals	:	are resolved within the absorption package

The cross section codes are summarized below. Refer to the table in section 5.9 of Ref.2 for more details and literature references.

Code	Model
0	Brown & Gould
1	Fireman (gas model)
2	Fireman (0.15 micron grains)
3	Fireman (0.60 micron grains)
4	as 0 plus EUV extension
5	Morrison & McCammon (recommended)

6.1.2 UV dust absorption

Calling sequence: CALL DERED(EE,VALIN,VALOUT,AV,IERR)

Argument EE : is wavelength in A
Argument VALIN : is input flux to be corrected
Argument VALOUT : is the returned corrected flux
Argument AV : is the A_V to be used (see below)
Argument IERR : is an error flag returned (see below)

To de-redden a measured value VALIN into the dereddened VALOUT, AV shall be supplied as a positive value.

To redden an unabsorbed value VALIN into an absorbed value VALOUT (as it is the case for fitting programs) AV shall be supplied as a negative value (this is taken care of internally in the routines provided here).

.cp4
The IERR flag is set to 1 if the following error occurred: the wavelength is outside the range 1000-10000 A for which the Seaton reddening empiric law is defined.

6.2 CURFIT interface

The CURFIT package is not part of the present package. The current implementation is strictly consistent with the code in Bevington (Ref. 8) with only minor changes. The user program generally calls directly only the CURFT and FCHI routines. These routines call internally the main function FUNC and the derivative routine FDER (which on its turn calls FUNC). In most cases the standard FDER in

Bevington is sufficient (in particular cases a time-optimized version, which avoids repeating calculations already done, may be useful).

All what is left to the user is to write his own function FUNC (the name is hardwired in CURFIT) according to the following calling sequence. Of course, if different fitting programs use different versions of FUNC, these versions shall reside in different libraries (or with the main). The user FUNC can do whatever he likes (just a plain call to a spectral form, or an integral, convolution, etc.). The calling sequence is :

```
Y = FUNC(X,I,A,NT)
```

where X is in line of principle an array (in our case of energies, frequencies or wavelengths) and I points to an element of this array. FUNC computes the function value at X (I). Generally in our case X is a scalar variable and I is therefore 1, but the array form is nevertheless included for compatibility with the CURFIT format.

A is the array of the fitted parameters, with NT elements (this latter information is generally unnecessary, but included again for compatibility with CURFIT format).

Instructions for building an user FUNC are given below in 7.3

The typical CURFIT application is arranged typically in a loop like the following, where some elements are left to the discretion of the user:

.cp3

```
FL=0.001
CL=large value

DOLOOP on niter iterations
  CALL CURFT
  (X,Y,SY,NPTS,NTERMS,MODE,A,DELTAA,SIGMAA,FL,YFIT,CHI)
  checks on FL
  FR=ABS(CL-CHI)
  IF(FR.LE.deltachisquare) exit loop
  CL=CHI
ENDLOOP
```

.cp5

The arguments of CURFT are described in Ref. 8. In our case MODE is always set to 1, which means weight on errors SY. The chi-square CHI is already reduced (divided by number of Dof).

The starting large value shall be therefore larger than any expected reasonable chi-square. A reasonable deltachisquare for convergence is 0.005, however if the number of points (hence the number of Dof) is very large a smaller value is requested (e.g. if one wants convergence to be reliable at a 68% confidence a value of 2.3/NPTS is adequate).

7. Customization

7.1 Parametric quantities

The following quantities are parametric, and as such their values are set in PARAMETER statements contained in INCLUDE members of the LUCIOMAC macro library (source in LUCIOMAC COPY). This way (see also 2.1) any change in the macro library will propagate in all programs, just by recompiling them without any edit. Most of such parameter are relevant to fitting programs only.

NCOMP Maximum number of components which can be handled together (added by SPESUM, see 4.3), currently set to 4

NFPARM Maximum number of parameters for one spectral form (currently 5, however no spectral form uses currently more than 4)

NCOMP and NFPARM are set in include member PARACP

MAXFIT Maximum number of parameter which can be fitted together. Currently set to 6 in include member PARAMF. Note that this is the value currently hardwired (i.e. not parametric) in CURFIT package.

.cp3

MAXENE maximum size of a response matrix (X-ray programs only) along energy direction. Current Exosat default is 272.

.cp3

MAXBIN maximum number of points in one spectrum (for X-ray programs maximum number of channels, also for response matrix). Current Exosat default is 133, and IUE default is 1500.

MAXENE and MAXBIN or only MAXBIN, as appropriate, shall be set in dedicated include members. As an example see

member PARAXX (fitting programs access it through member PARAXR), which is suitable for X-ray programs in the Exosat system or member PARAUU (fitting programs access it through member PARAUV) for UV programs.

.cp5

NITER is the maximum number of iterations for the CURFIT calling loop. This is set to 100 in include members PARAXR and PARAUV; build your own for other purposes).

If one wants to use different values than indicated above, two policies are possible. One (for officially supported programs) is to edit the source file LUCIOMAC COPY, change or add new include members, and regenerate the library with the command **LUCIOMAC** (this generates afresh the LUCIOMAC MACLIB).

Otherwise one may create a MACLIB of his own (see Ref. 7 and 6), and, when (re)compiling the various programs, search it first (it is enough to issue the CMS command in the form :

GLOBAL MACLIB user-library **LUCIOMAC**

7.2 Overriding a spectral form or defining new ones

There are currently 4 codes (9 and 13 to 15) available for new spectral forms. If more than 15 forms are desired, it is necessary to edit the source of most distribution routines, and change the value in common SPECMN. New spectral form may be defined either officially (replacing one of the spare ones in the library), or privately.

In the latter case one has just to create a Fortran routine named SPEnn (nn=9,13,14,15) according to the rules here indicated. If one wants instead to alter one of the existing forms, one will call such routine with the same name as the existing form. In the simplest case, the new routine will reside in a file SPEnn FORTRAN, compiled as SPEnn TEXT (advanced users may want to create their private library).

Provided the TEXT file has the same name as the routine, the CMS LOAD command will automatically find it, and use it instead of the one in the LUCIOLIB library. This applies both if LOAD is done at run time, or before generating a module with GENMOD (i.e. applies both to TEXT and MODULE files, invoked

with EXECUTE or generated with COMLINK ; see Ref. 6 for details on the two latter commands).

.cp3
Using private libraries, or the CMS INCLUDE command to relocate TEXT files of arbitrary name, shall be obvious to advanced users (see again Ref. 6).

A template for a standard SPEnn is provided in the source file of the package. All what one has to do is :

To replace all occurrences of SPEnn with the wished name

Optionally, to enable an alias name, and replace all occurrences of the word "alias" with such name.

To set NPARM=n to the number of parameters for the new form.

To set NAME to the spectral form name, and the array PARAM to the names of the single parameters.

To replace the INFOnn name with the wished name

To insert the Fortran code for the calculation at or before the line labeled "Y=function"

To decomment all other statements (prefixed with *)

If the new approach is followed concerning unit conversion, one can decomment also the statements prefixed by C****, provided the calculation is done in dN/dE.

The new function is then ready to be compiled, and used as indicated above.

7.3 Building a FUNC for CURFIT

A CURFIT-format function FUNC (see 6.2 above) may be built as follows:

First statement is FUNCTION FUNC(X,I,A,NT)
Argument arrays X and A are X(*) and A(*)
Communication of other parameters not in the argument list may occur through dedicated common blocks (e.g. the response matrix for X-ray programs).
If wished the actual calculation may be performed only when

the values of the parameter in A have actually changed (this is worthwhile for complex or long calculations, e.g. response matrix convolution).
FUNC shall be assigned the wished value

In the simplest cases it is enough to equate FUNC with one of the distribution routines (see 4. above), like SPESUM (x,A), SPECS(x,A), SPEX2(x,A) etc., where x is X(I).

.cp3
In more complex cases (e.g. convolution, integration etc.), the distribution routines will be called, and their result used in the calculation.

7.4 Building aliases

As noted above (in section 4 and 6.2) most functions are of the form either f(x,A) or f(X,I,A,NT). However cases exist when a form f(x), the function of a single variable, is desired (e.g. most integration routines require the function to be integrated as an argument, and such function shall be of this format).

.cp4
The above requirement is met building an alias of an existing distribution routine. Other aliases may be required if one wants fluxes in units different from the standard three cases used in the package (this either e.g. to have keV/cm^2/s/keV vs keV, or to provide scaling).

A case of simple alias, like converting dN/dE into keV/cm^2/s/keV, is obtained with a statement like :

```
ESPE(E) = E * SPEX2(E,A)
```

(this is here shown as a statement function; this cannot be passed as argument, therefore a proper external function ESPE (E) shall be built around the single statement :

```
ESPE = E * SPEX2(E,A)
```

Of course the array A is not known to the function, and shall be passed through a common block. The same case if one wants dN/dE in f(E) form.
The following Fortran code will do :

```
REAL FUNCTION DNDE(E)
```

```
      common block containing A
      DNDE = SPEX2(E,A)
      RETURN
```

Note that the example here is built around SPEX2 (which handles internally the difference between fitted and fixed parameters), but the same may apply also to the single-form routine SPECS. The distribution routines SPECSX and SPECUV (4.2.2/3) are already in the f(x) form.

.cp53

References

1. Chiappetti L. & Bellavita R. "Un sistema per il trattamento dei dati IUE - Manuale per l'uso", Rapporto interno LFCTR, June 1981

2. Chiappetti L. & Garilli B. "The IFCTR Exosat analysis system - Users' handbook", Rapporto interno IFCTR, January 1986

3. Chiappetti L., "Un chiarimento sulle forme spettrali", Informal note, not dated (January 1986 ?)

4. Chiappetti L., "Formule di conversione per i flussi spettrali e altre tabelle", Informal note, July 1982

5. Belloni T. & Chiappetti L., "Sistema di analisi IUE - Manuale per l'uso", To be issued (December 1987).

6. Chiappetti L., Programmers' Productivity Tools, to be written (refer to Ref. 2 section 2.3 in the meanwhile)

7. IBM manual: VS Fortran Programming Guide (Release 4.0), SC26-4118-0 (particularly pag. 230 ff)

8. Bevington P.R., "Data Reduction and Error Analysis for Physical Sciences", McGraw-Hill, 1969

9. Chiappetti L., The new user-program communication package, IFC software documentation, August 1987

Acknowledgments

Some of the spectral form routines have been provided in original form by T.Belloni. Useful discussions with A.L.Ciapi, T.Belloni and B.Garilli are acknowledged.

Appendix: formulae for spectral forms

In all formulae y denotes a generic flux unit, x a generic energy unit (wavelength, frequency or energy), C the normalization factor, U a generic universal constant (whose value is indicated case by case), and A_i the i-th parameter of the spectral form.

N.1: Power law $y = Cx^{-a}$

x = l,n,E
y = F_l,F_n,dN/dE
A_1 = C
A_2 = a

Calculation done in any unit

N.2: Blackbody

$F_l = CU_l l^{-5} / (e^{E/kT} - 1)$
$F_n = CU_n n^3 / (e^{E/kT} - 1)$
$dN/dE = CU_E E^2 / (e^{E/kT} - 1)$

A_1 = C
U_l=U_n=U_E=1 if PHYSIC=.TRUE. (current for Exosat and IUE), otherwise
U_l=3.7 10^{27}
U_n=4.6 10^{-47}
U_E=9.9 10^{31}

A_2 = kT (if in keV, current for Exosat)
A_2 = T (if in K , current for IUE)

.cp16

N.3: Bremsstrahlung $F_1 = CU_1(kT)^{-1/2} e^{-E/kT} 0.8 (kT/E)^{0.4} 1^{-2}$

$F_n = CU_n(kT)^{-1/2} e^{-E/kt} 0.8 (kT/E)^{0.4}$

$dN/dE = CU_E(kT)^{-1/2} e^{-E/kT} 0.8 (kT/E)^{0.4} E^{-1}$

$A_1 = C$

$U_1=U_n=U_E=1$ if PHYSIC=.TRUE. (current for Exosat and IUE), otherwise$U_1=2.4\ 10^{-27}$

$U_n=8.2\ 10^{-46}$

$U_E=1.2\ 10^{-19}$

$A_2 = kT$ (if in keV, current for Exosat)

$A_2 = T$ (if in K , current for IUE)

.cp7

N.4: Exponential $F_1 = CU_1(kT)^{-1/2} e^{-E/kT} 1^{-2}$

$F_n = CU_n(kT)^{-1/2} e^{-E/kt}$

$dN/dE = CU_E(kT)^{-1/2} e^{-E/kT} E^{-1}$

A_1,A_2 etc. : same definition as form n.3

N.5: Bremsstrahlung $dN/dE = Ce^{-E/kT} E^{-1} (g_1+0.34g_2)$ E < 20 keV

$Ce^{-E/kT} E^{-1} g_3$ E > 20 keV

g_1 = GAUNT(E,kT,Z=1)

g_2 = GAUNT(E,kT,Z=2)

GAUNT according to Kellogg, Baldwin & Tucker

$g_3 = 0.9 (kT/E)^a$

$a = (0.3\ kT/E)^{0.15}$

$A_1 = C$

$A_2 = kT$ (if in keV, current for Exosat)

A_2 = T (if in K , current for IUE)

Calculation done in dN/dE, than converted to other units

N.6: Boltzmann $dN/dE = CE^{-a}\ e^{-E/kT}$

A_1 = C
A_2 = kT (if in keV, current for Exosat)
A_2 = T (if in K , current for IUE)
A_3 = a

Calculation done in dN/dE, than converted to other units

N.7: Bath thick disk y = C $INT1,a; 2pr B(T(r))dr

B(T) is a blackbody (form N.2) with unit normalization

The temperature profile $T(r)=T_\star r^{-0.75}\ (1-r^{-1/2})0.25$

the radius r is in units of the inner radius R_{in}

A_1 = C
A_2 = $kT_\star$ (if in keV, current for Exosat)
A_2 = $T_\star$ (if in K , current for IUE)
A_3 = a = R_{out}/R_{in}

Calculation is done in any unit, make reference to form n.2 for the way the blackbody is calculated

N.8: Gaussian y = C exp $OQ -1/2 $OR(x-$x_{peak}$)/s$CR ^{2}$CQ

A_1 = C
A_2 = x_{peak}
s = 0.4246 A_3
A_3 is FWHM

Calculation is done in any unit

N.10: Compton $dN/dE = C\ (E/kT)^2\ e^{-e/kT}\ \$INT0,@;e^{-r}f(r)dr$

$f(r) = r^{n-1}\ \$OQ1+rkT/E\$CQ^{n+3}$

$n = (2.25 + g)^{1/2} - 1.5$

$g = 511\ p^2\ /\ [\ 3kT(A_3 + 2/3)^2]$

A_1 = C
A_2 = kT (if in keV, current for Exosat)
A_2 = T (if in K , current for IUE)

Calculation done in dN/dE, than converted to other units

N.11: Czerny modified blackbody

$$dN/dE = B(E,T)\ [a/(a+0.4)]^{1/2}$$

B(E,T) is a blackbody (form n.2), to which parameters A_1 and
A_2 refer.

$a = 2.63\ A_3\ (kT)^{-3.5}\ (1-e^{-E/kT})\ (E/kT)^{-3}$

Calculation done in dN/dE, than converted to other units

N.12: Twin power law $y = C_1x^{-a}$ $x < A_4$
C_2x^{-b} $x > A_4$

$A_1 = C_1$
$A_2 = a$
$A_3 = b$
A_4 = break position

$C_2 = A_1A_4^{b-a}$

Calculation done in any unit

We remind that the distribution function SPESUM is the sum of a number of components selected among the above; the X-

ray and UV distribution functions SPEX2 and SPEUV2 are the result of SPESUM, corrected for interstellar absorption.

The function FUNC used by the Exosat (X-ray) FIT and GRID programs is the one contained in FIT FORTRAN and GRID FORTRAN (**not** the one in MELIB LIBRARY which is now obsolete). The dN/dE spectrum at energy E_j, $S(E_j)$ is computed using SPEX2. The convolved function, counts in the i-th channel, is:

$$C_i = \$SUM_j; R_{ji} S(E_j)$$

where R_{ji} is the response of channel i to a photon at energy E_j.

The function FUNC used by the IUE (UV) UVFIT and UVGRID programs is contained in IUEAUXLB LIBRARY. According to whether one selects the monochromatic flux or pseudo-monochromomatic flux (for wide bins), it corresponds either to:

the F_l computed using SPEUV2 or to

the integral $\$INT_{l_i-Dl_i,l_i+Dl_i}; F_l dl$